młody TECHNIK

InforMik

MAGAZYN KOMPUTEROWY „MŁODEGO TECHNIKA"

I
1988



Nasz komentarz:

O POLIGLOTACH I ŚWIĘTOSZKACH

Miałem niedawno przyjemność wysłuchać wywodów pewnego skądinąd biegłego i zasługującego na szacunek informatyka. Na kursie, przeznaczonym dla początkujących użytkowników mikrokomputerów wypowiedział się on z pogardą o wszelkim spolszczonym oprogramowaniu, a zwłaszcza polskojęzycznych podręcznikach do zagranicznych programów. Nie ma rady: użytkownik komputera musi znać angielski, jak niegdyś polski szlachciura — łacinę. Kwestia terminologii jest drugorzędna. Mamy oto np. angielski termin: *typed constant*. Jak nazwać go po polsku? Proste: typowana konstanta! Jeśli kandydaci na użytkowników komputerów odważą się mieć odmienne zdanie i zatęsknią za programem, który zagada do nich po polsku, będzie to tylko dowodem ich cywilizacyjnej degeneracji.

Nie dajmy się zwariować! Jeżeli ktoś oferuje nam obcojęzyczną dokumentację i obcojęzyczny program zachwalając ich oryginalność, i twierdzi, że dla własnego dobra powinniśmy nauczyć się języka oryginału, to czym prędzej rozejrzyjmy się za innym partnerem. Komputery są bowiem dla ludzi, w tym także tych nie znających języków obcych albo nie władających nimi dość biegle.

Przedstawiony powyżej problem posłużył jedynie za pretekst do zajęcia się innym, nie mniej poważnym, chociaż odmiennej natury. Dotyczy on mianowicie przeróbek i adaptacji oprogramowania pochodzenia zagranicznego. Można spotkać poglądy, jakoby samo korzystanie z oryginalnego, nie przerobionego oprogramowania i oryginalnej dokumentacji mieściło się jeszcze w granicach etycznej tolerancji, ale jakiekolwiek ich przeróbki, adaptacje i tłumaczenia zasługiwały na całkowitą dezaprobatę.

Przyznaję, że wewnętrznie podzielam poglądy osób, występujących w obronie praw autorskich twórców oprogramowania zagranicznego w naszym kraju. Nie ukrywam jednak, że cała gorąca dyskusja, jaka ostatnio wybuchła na ten temat na łamach niektórych periodyków informatycznych, mocno mnie mierzi. W moim odczuciu jest ona bowiem kolejnym objawem typowo polskiej przywary, jaką jest maniacka skłonność do naprawiania świata gadaniem. Uważam bowiem, że jeśli spieramy się o pryncypia, to przynajmniej szczerze, że nieetyczne jest nie tylko „grzebanie" w cudzym programie, ale i nielicencjonowane korzystanie z tego oprogramowania w wersji oryginalnej.

Postawmy sprawę wprost. Korzystanie w jakiejkolwiek formie z cudzego dorobku intelektualnego bez zgody autora i bez ewentualnej rekompensaty jest nieetyczne — to nie ulega wątpliwości. Z drugiej strony fakt korzystania z zagranicznego oprogramowania w Polsce jest faktem, którego nie da się zwalczyć umoralniającymi pogawędkami. Co więc jest w takiej sytuacji bardziej etyczne: sprzedaż oryginalnego oprogramowania z oryginalną dokumentacją, do czego nikt w Polsce palca w sposób twórczy nie przyłożył, czy też sprzedaż tego samego oprogramowania, ale spolszczonego, albo przynajmniej adaptowanego do specyfiki języka polskiego, no i oczywiście z przyzwoitą dokumentacją po polsku? W tym drugim przypadku można zresztą sprawę postawić inaczej: obiektem sprzedaży nie jest sam program zagraniczny, który przecież da się bez większego trudu skopiować tu i ówdzie, ale właśnie polska dokumentacja i dokonane w programie przeróbki, które w końcu decydują o przydatności programu dla polskiego użytkownika.

Zdaję sobie sprawę, że przytoczonym tu poglądom daleko jest do moralnej czystości, lecz pragnę zwrócić uwagę, że działalność gospodarcza z reguły nie bywa szkołą etyki — tak jest niestety na całym świecie. Etyka jest respektowana w świecie tam, gdzie na jej straży stoją pospołu egzekwowane prawo oraz etos silnego środowiska. W Polsce brak nam dziś obu tych elementów.

Z obiektywnych okoliczności wynika, że jesteśmy skazani na korzystanie z cudzego dorobku, nie mogąc jak na razie zaoferować światu wiele w zamian. Skoro tak, to korzystajmy z tego dorobku jak najefektywniej. Jest zaś rzeczą oczywistą, że spolszczony program z polskojęzyczną dokumentacją jest nieporównanie bardziej użyteczny, niż program oryginalny z lichą kserokopią anglojęzycznego podręcznika.

Sprawa jest prosta: musimy wybierać między komfortem czystego sumienia a naszym narodowym interesem, który nakazuje jak najszerzej korzystać z obcego dorobku intelektualnego, aby dźwigać się z gospodarczego i technologicznego upadku. Ostatnich kilkaset lat historii wyraźnie wskazuje, że moralna słuszność nie musi iść w parze z osiągnięciami w różnych dziedzinach życia praktycznego, w tym gospodarczego. Mało kto dziś pamięta, że nawet gospodarczy cud Japonii zaczął się od niewolniczego wręcz kopiowania cudzych osiągnięć technicznych.

Można w naszej gospodarce wskazać mnóstwo rzeczy jeszcze mniej moralnych, niż nierespektowanie praw autorskich obywateli państw obcych. Niezwracanie długów, zaciągniętych niegdyś na określonych, zaakceptowanych w swoim czasie warunkach, także nie jest moralne. Uważam jednak, że powinniśmy się wprawdzie z tego powodu rumienić ze wstydu, ale mimo to spłatę zadłużenia podporządkować raczej naszemu narodowemu interesowi, niż racjom moralnym. Co do zagranicznych programistów można śmiało założyć, że tworząc swe dzieła nie spodziewali się zarobić na nas ani centa, zaś godziwą satysfakcję materialną w twardej walucie otrzymali już ze sprzedaży swoich programów na rynkach zachodnich. Zastanówmy się przy tym, czy świat, wobec którego staramy się być tak bardzo fair, był i jest fair także wobec nas samych.

Są oczywiście reguły, których naruszać w moim przekonaniu nie wolno w żadnym przypadku, np. zacierać informacji o twórcach przerabianego programu. Za głęboko nieetyczne uważam też nieautoryzowane kopiowanie dzieł programistów krajowych. Bynajmniej nie dlatego, że metodą Kalego dzielę programistów na „swoich" i „obcych". Po prostu w przypadku programów pisanych na rynek krajowy autor może liczyć tylko na dość skromne wpływy ze sprzedaży swych dzieł w Polsce. Krzywda wyrządzona autorowi jest więc tutaj nieporównanie większa.

Wiem, że przedstawionymi tu poglądami narażę się wielu polemistom, zwłaszcza że pewne sprawy celowo przerysowałem. Mimo to proponuję jednak, żeby wreszcie przestać biadolić, postulować i nawoływać, zaś w zamian zakasać rękawy i zabrać się do roboty — do jakiejkolwiek roboty, dającej praktyczne korzyści. Uważam bowiem, że autentyczna praca, jeśli nawet czasami bywa na bakier z niektórymi normami szeroko rozumianej etyki, jest sama w sobie bardziej moralna niż najbardziej nawet umoralniające gadulstwo.

Roland Wacławek

CIEKAWE KSIĄŻKI

Donald Hearn, M. Pauline Baker: **Mikrokomputernaja grafika**, Moskwa 1987; tytuł oryginału: Microcomputer Graphics. Techniques and Applications, rok wydania 1983. Na polskim rynku wydawniczym często pojawiają się doskonałe pozycje w języku rosyjskim będące udanymi tłumaczeniami książek wydawanych na Zachodzie. Wśród nich znajdują się również liczne publikacje dotyczące komputerów i informatyki. Do zakupu zachęca nie tylko dobra strona edytorska, ale i bardzo niska cena, z reguły trzy-, czterokrotnie niższa od ceny analogicznych pozycji w języku polskim.

Książka „Mikrokomputernaja grafika" omawia wiele zagadnień związanych z przedstawianiem obrazów na ekranie komputera począwszy od najprostszych, a skończywszy na takich, które wymagają opanowania obszernego aparatu matematycznego. W pierwszej części książki autorzy omawiają ogólne problemy grafiki mikrokomputerowej przytaczając liczne przykłady amatorskich i profesjonalnych jej zastosowań. Następna część to podstawy grafiki. Autorzy rozpoczynają ją od przedstawienia sposobów rysowania linii na ekranie przy wykorzystaniu instrukcji PLOT, a kończą na tworzeniu wykresów dowolnych funkcji. Część trzecia została poświęcona metodom graficznym o średnim stopniu złożoności. Tutaj Czytelnik może znaleźć m.in. informacje o przekształceniach pła-

Dalszy ciąg na s. 8

młody TECHNIK

InforMik

MŁODY TECHNIK — INFORMIK
MAGAZYN KOMPUTEROWY
MŁODEGO TECHNIKA
NR 1(5) ROCZNIK II

SPIS TREŚCI

Zdjęcia w numerze: Władysław P. Jabłoński, Jacek Nowicki, Grzegorz Zalot, ze zbiorów redakcji.
Rysunki: Roman Gaik.

ALL RIGHTS RESERVED

Powszechnie wiadomo, że jeśli idzie o języki obce, to my, Polacy, jesteśmy uzdolnieni, ale rzadko kto z nas mówi bezbłędnie np. po angielsku czy niemiecku. Dawno, dawno temu Polacy słynęli z perfekcyjnej znajomości francuskiego — dziś i z tym jest nie najlepiej. A szkoda, bo znajomość innych języków to rzecz bardzo przydatna i chyba nie muszę nikogo o tym przekonywać...

Współczesny świat to świat przenikających się kultur, idei, światopoglądów, to także świat, w którym używamy leków ze Szwajcarii, odżywek z RFN, samochodów z Japonii, aparatów fotograficznych z NRD itp. itd. Nic też dziwnego, że pojawia się nagle przed nami problem korzystania z tych wszystkich dóbr i to korzystania w sposób zgodny z ich przeznaczeniem i możliwościami, co umożliwia osiąganie właściwych rezultatów używania i przedłuża wydatnie życie danego urządzenia.

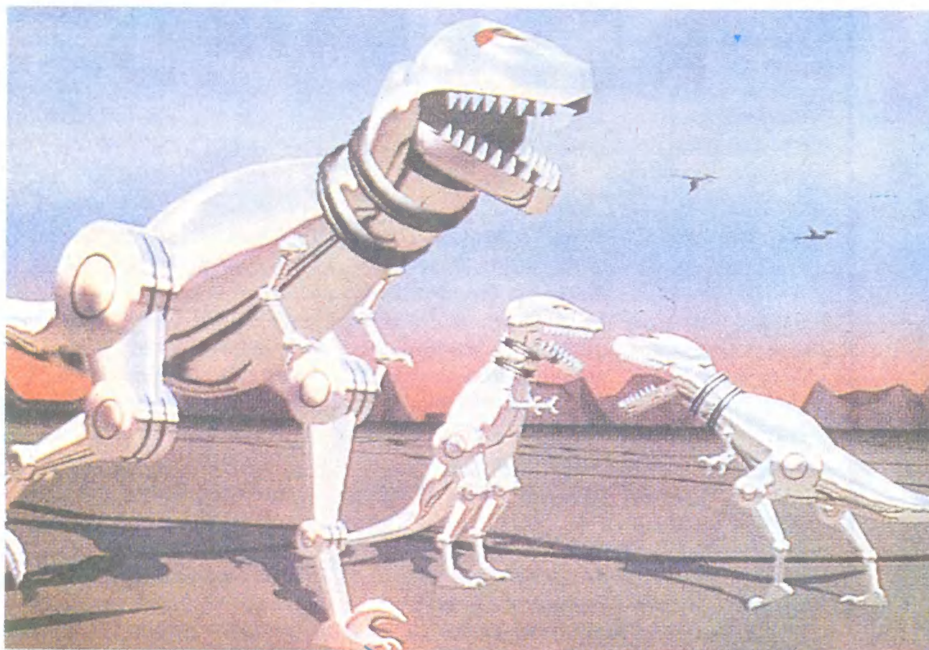
Jest chyba niewiele dziedzin, które tak wymagałyby owej dokładności i właściwego wykorzystania, jak mikroinformatyka. Wielu naszych Czytelników wie dokładnie jak kończy się wpisanie do mikrokomputera programu, w którym brakuje jednego znaku. Dlatego też uważam, że problem właściwego zrozumienia pola i zakresu możliwości programu użytkowego jest — moim zdaniem — problemem pierwszorzędnym. Program, którego instrukcji nie rozumiemy, nie jest w pełni wykorzystywany, a z tym naszym angielskim nie zawsze jest najlepiej. I tu dochodzimy do sedna problemu...

Czy opracowywanie zachodnich programów użytkowych jest kradzieżą? Czy osoba tłumacza rzeczywiście nic dla polskiego użytkownika nie wnosi? Na ile czyn nie ścigany prawem można nazwać przestępstwem? Są to wszystko pytania, na które trzeba sobie odpowiedzieć w kraju, gdzie prywatnie ludzie bardziej garną się do wiedzy i postępu niż czynią to oficjalne instytucje w tym celu powołane.

Piszę na ten temat już po raz drugi, bo ciągle budzi on wiele kontrowersji. Pierwszy felieton poświęciłem aspektowi moralnemu całej sprawy, dziś chciałbym zwrócić uwagę na jej stronę ekonomiczno-użytkową. Otóż — jak mi się wydaje — szeroko rozumiana komputeryzacja jest jedyną szansą dla naszego przemysłu. Tam, gdzie ludzie pracują źle i niewydajnie, sterowana komputerem automatyzacja jest jedynym wyjściem. Tam, gdzie dziesiątki urzędników przemieszczają tony papierów bez zadowalających rezultatów, komputeryzacja jest warunkiem sine qua non. Jednak same te zmyślne maszynki problemów naszych nie rozwiążą — potrzeba im odpowiedniego oprogramowania. Nie zawsze urzędnicy czy technicy obsługujący komputer znają na przykład język angielski. Czy w takim razie lepiej szkolić ich w angielskim czy dać im po prostu przetłumaczony program?

Niektórzy puryści etyczno-moralni branży mikrokomputerowej przekonali mnie już prawie, że tłumaczenia Manna, Gogola, Hemingwaya czy Conan-Doyle'a są naruszeniem praw autorskich. Ciekaw jestem co na to prawo i tłumacze? Już widzę — in my mind's eyes, natürlich — jak stosowni funkcjonariusze wloką przez miasto zakutego w kajdany tłumacza „Ulissesa" na miejsce kaźni, a pewien mój znajomy tłumaczy nielegalnie instrukcje lotnicze i rozprowadza wśród pilotów po czarnorynkowych cenach. Sam — choć poliglotą nie jestem — obiecuję sobie już dziś, że opłaty za pokątne tłumaczenia napisów na opakowaniach leków będę pobierał wyłącznie w papierkach z nadrukami w językach tłumaczonych, bo pomysł banknotów jest chiński, ale na Zachodzie wykupili podobno prawa autorskie...

JERZY KLAWIŃSKI



BOGUSŁAW KOSSAKOWSKI
STANISŁAW TRZCIŃSKI

GRAFIKA TIMEX-a 2048

Komputer TIMEX 2048, poza lepszą klawiaturą, różni się od ZX Spectrum znacznie większymi możliwościami graficznymi (ekran o rozdzielczości 512×192 punkty). Niestety większość nabywców mikrokomputerów TIMEX 2048 wskutek braku opracowań firmowych nie jest w stanie korzystać z tych możliwości.

Mikrokomputer ten posiada cztery tryby graficzne, które kolejno omówimy.

TRYB 1

Tryb ten ustawiany jest automatycznie po włączeniu zasilania. Działają w nim wszystkie rozkazy graficzne BASIC-a ZX Spectrum. Z innych trybów wraca się do trybu 1 komendą OUT 255,0. Obraz na ekranie wyświetlany jest w tym trybie z tzw. obszaru pamięci pierwszego ekranu od adresu 16384 do 22527 (szesnastkowo: #4000 do #57FF). Atrybuty ekranu zajmują obszar o adresach 22528 do 23295 (szesnastkowo: #5800 do #5AFF). Można w nim definiować atrybuty znaku 8×8 pikseli. Działają w nim wszystkie rozkazy graficzne BASIC-a ZX Spectrum.

TRYB 2

Tryb ten włączany jest komendą OUT 255,1. Na ekranie wyświetlana jest wtedy zawartość pamięci drugiego ekranu tj. obszar o adresach 24576 do 30719 (szesnastkowo: #7800 do #7AFF). W trybie tym ekran posiada również rozdzielczość 256×192 punkty, a atrybuty odnoszą się do znaku 8×8. Po włączeniu komputera i wykonaniu komendy OUT 255,1 otrzymujemy czarny ekran zaś przyciskanie jakichkolwiek klawiszy nic na nim nie zmieni. Jedynie wykonanie komendy OUT 255,0 spowoduje powrót do trybu 1. Dzieje się tak dlatego, że wszystkie bajty atrybutów mają wartość 0, co oznacza czarny papier i czarny atrament. Jeśli wpiszemy POKE-ami w pole atrybutów drugiego ekranu liczby 56 otrzymamy biały ekran.

W trybie tym bez specjalnego oprogramowania trudno jest cokolwiek wykonywać, gdyż nawet napisy umieszczane w tzw. oknie systemowym nie są na drugim ekranie wyświetlane. Jednym z wyjść może być tworzenie grafiki na pierwszym ekranie i następnie przepisywanie jej do drugiego ekranu. Bezpośrednie umieszczenie punktu o współrzędnych X,Y na drugim ekranie wymaga obliczenia adresu bajtu, w którym znajduje się ten punkt oraz obliczenia położenia bitu w tym bajcie. Ponadto należy pamiętać, że pamięć drugiego ekranu leży w obszarze zajmowanym przez programy w BASIC-u. Szczegółowo zagadnienia te zostaną omówione w opisie trybu 4.

TRYB 3

W trybie 3 wyświetlany jest obszar pamięci pierwszego ekranu. Natomiast atrybuty zajmują obszar pamięci drugiego ekranu. Daje to możliwości definiowania atrybutów dla poszczególnych bajtów. Oznacza to, że każdy „rządek" z ośmiu pikseli może mieć inny kolor atramentu i tła, migać, mieć normalną lub podwyższoną jasność. Tryb ten włącza się ko-

mendą OUT 255,2. Można w nim posługiwać się częścią rozkazów graficznych BASIC-a, jak PLOT, DRAW i PRINT oraz wyświetlać listing programu pod warunkiem, że do każdego bajtu pamięci drugiego ekranu wpisze się odpowiednie atrybuty. Aby otrzymać np. czarne litery na białym tle należy do wszystkich komórek pamięci drugiego ekranu wpisać 56. W trybie tym nie działają rozkazy BASIC-a: PAPER, INK, BRIGHT, FLASH. Należy przy tym pamiętać, że w trybie tym atrybuty znajdują się w obszarze zajmowanym przez programy w BASIC-u.

TRYB 4

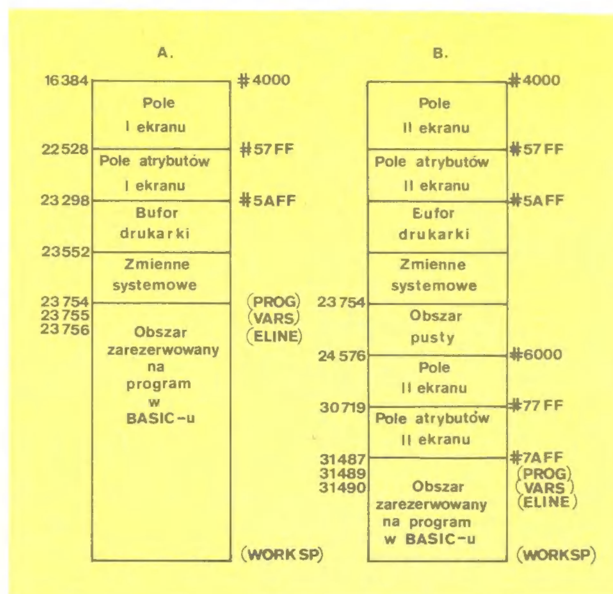
Tryb ten włącza się komendą OUT 255,6. Uzyskuje się w nim grafikę o podwójnej rozdzielczości w poziomie, tj. 512×192 punkty. Kolory tła i tuszu są w tym trybie identyczne dla wszystkich punktów ekranu. Po komendzie OUT 255,6 otrzymuje się białe tło i czarny tusz. Można uzyskać inne kombinacje kolorów wysyłając do portu 255 liczby według zamieszczonej obok tabeli barw. Atrybuty można definiować tylko przy włączaniu trybu.

TABELA BARW

Wartość wysyłana do portu 255	Kolor tuszu	Kolor tła
6	Czarny	Biały
8+6	Niebieski	Żółty
16+6	Czerwony	Błękitny
24+6	Fioletowy	Zielony
32+6	Zielony	Fioletowy
40+6	Błękitny	Czerwony
48+6	Żółty	Niebieski
56+6	Biały	Czarny

Do programowania grafiki w tym trybie niezbędna jest ponadto znajomość mapy pamięci, informacja o tym jak tworzony jest obraz, jak obliczyć adres bajtu, w którym znajduje się punkt, położenie bitu w tym bajcie i w końcu jak szybko obraz na ekranie skasować.

W trybach 2, 3 i 4 na pamięć ekranu i atrybutów zajęte są dwa obszary pamięci. Obszary te przedstawiono na części mapy pamięci na rys. 1. Na mapie tej widać, że pamięć drugiego ekranu i jego atrybuty leżą w obszarze zajmowanym przez programy w BASIC-u. Może to prowadzić do niszczenia obrazu lub, co gorsza, do niszczenia programu przez punkty obrazu i w konsekwencji do „zawieszenia" się systemu. W związku z tym konieczne jest przesunięcie początku programu w BASIC-u poza obszar atrybutów drugiego ekranu. Należy w tym celu w zmienne systemowe PROG i VARS wstawić POKE-ami liczbę 31489, w zmienną E-LINE — 31490, natomiast w bajt o adresie 31489 — liczbę 128. Przesunięcia początku programów w BASIC-u wygodniej jest dokonywać za pomocą krótkiego programu w BASIC-u. Po tym zabiegu można już ręcznie lub z taśmy instrukcją LOAD wpisywać programy na obliczanie adresów i rysowanie punktów.

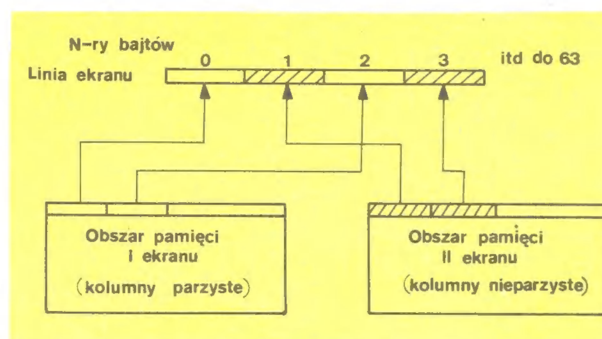


Rys. 1. Część mapy pamięci: a — w trybie 1; b — w trybach 2, 3 i 4. Zmienne (VARS) i (ELINE) przed wpisaniem programu w BASIC-u

Do pisania takich programów konieczna jest znajomość tego, w jaki sposób powstaje w trybie 4 obraz na ekranie. Przedstawiono to na rysunkach nr 2 i 3. Kolumny parzyste wyświetlane są z obszaru pamięci ekranu pierwszego, a kolumny nieparzyste z obszaru pamięci drugiego ekranu. Kolejność wyświetlania linii pozostaje taka sama jak w ZX Spectrum. Natomiast kolejne bajty do wyświetlania w linii ekranu pobierane są na przemian raz z jednego, raz z drugiego obszaru pamięci (rys. 2). W ten sposób powstaje obraz składający się z 64 kolumn (rys. 3).

Aby obliczyć położenie w pamięci obrazu adresu punktu o współrzędnych X,Y należy w pierwszym rzędzie ustalić czy punkt ten znajduje się w kolumnie parzystej czy nieparzystej, tj. czy leży on w obszarze pamięci pierwszego czy drugiego ekranu. Odpowiadające sobie adresy bajtów pamięci pierwszego i drugiego ekranu różnią się o 8192. Do obliczania adresu punktu można zastosować taką samą zależność dla kolumn parzystych i nieparzystych, a następnie po stwierdzeniu, że punkt leży w kolumnie nieparzystej

Rys. 2. Powstawanie linii obrazu w trybie 4 (w grafice 512 na 192 punkty)



```
   1 REM Program "LD4.1"
   5 REM BOGUSLAW KOSSAKOWSKI.   STANISLAW TRZCINSKI 87.06.01
  10 REM Ten program przesuwa poczatek programu w Basicu za obsza
r atrybutow drugiego ekranu
  20 POKE 23635,1: POKE 23636,123: POKE 23627,1: POKE 23728,123:
POKE 23641,2: POKE 23642,123: POKE 31489,128
```

```
   1 REM Program "RF4.1"
   2 REM BOGUSLAW KOSSAKOWSKI. STANISLAW TRZCINSKI 87.06.01
   5 REM Ten program rysuje wykres prawie kazdej funkcji w zakres
ie -s do s w grafice o podwojnej rozdzielczosci
  10 REM Kasuje obszar pamieci drugiego ekranu ale bardzo wolno
  20 REM Rysowanie osi x
  30 LET k=0
  40 FOR f=0 TO 511 STEP 8
  50 LET y=96
  60 GO TO 220
  70 REM Rysowanie osi y
  80 LET k=1
  90 FOR y=0 TO 191
 100 IF y=96 THEN  NEXT y
 110 LET f=255
 120 GO TO 220
 130 REM Rysowanie wykresu funkcji dla zmiennej x=-s do s
 140 LET k=2
 150 INPUT "s=";s,  "e$=";e$
 160 PRINT "-s<=x<=s          y=";e$;""
 170 PRINT "s=";s
 180 FOR f=0 TO 511
 190 LET x=(f-256)*s/256: IF x=0 THEN  NEXT f
 200 LET y=96+INT ((VAL e$)+.5)
 210 IF y>191 OR y<0 THEN  GO TO 350
 220 LET a=16384+32*(INT ((191-y)/8)-INT ((191-y)/64)*8+8*(191-y-
INT ((191-y)/8)*8)+64*INT ((191-y)/64))+INT (f/16)
 230 LET c=8192*(INT (f/8 )/2>INT (INT (f/8)/2))
 240 LET b=1+f-INT (f/8)*8
 250 DIM d(8)
 260 LET d(1)=128: LET d(2)=64: LET d(3)=32: LET d(4)=16: LET d(5
)=8: LET d(6)=4: LET d(7)=2: LET d(8)=1
 270 IF k=0 THEN  LET d(b)=255
 280 LET n=d(b)+PEEK (a+c)
 290 IF n>=255 THEN  LET n=255
 300 POKE a+c,n
 310 IF k=0 THEN  NEXT f
 320 IF k=0 THEN  GO TO 80
 330 IF k=1 THEN  NEXT y
 340 IF k=1 THEN  GO TO 140
 350 NEXT f
 360 STOP
 370 REM Kasowanie pamieci ekranow
 380 CLS
 390 FOR k=24376 TO 30719
 400 POKE k,0
 410 NEXT k
```

```
   1 REM Program "LD4.2"
   2 REM BOGUSLAW KOSSAKOWSKI,   STANISLAW TRZCINSKI 87.06.01
   5 REM Ten program  wpisuje program w kodzie maszynowym i przes
uwa poczatek programu w Basicu.
  10 CLEAR 50000
  20 FOR i=50001 TO 50074
  30 READ a
  40 POKE i,a
  50 NEXT i
  60 DATA 62,0,33,0,96,119,17,1,96,1,255,23,237,176,201,58,253,95
,71,42,254,95,203,28,203,29,77,203,29,203,29,203,29,31,55,31,167,
31,168,230,248,168,87,121,7,7,7,168,230,199,168,7,7,95,58,254,95,
230,7,103,62,8,148,71,175,55,23,16,253,103,26,180,18,201
  70 POKE 23635,1: POKE 23636,123: POKE 23627,1: POKE 23628,123:
POKE 23641,2: POKE 23642,123: POKE 31489,128
```

```
   1 REM Program "RF4.2"
   5 REM BOGUSLAW KOSSAKOWSKI,  STANISLAW TRZCINSKI 87.06.01
  15 REM Ten program rysuje wykres prawie kazdej funkcji w trybie
 o zwiekszonej rozdzielczosci.
  25 REM Rysowanie osi X
 100 FOR x=0 TO 511
 200 POKE 24573,96
 300 POKE 24575,INT (x/256): POKE 24574,x-256*INT (x/256)
 400 RANDOMIZE USR 50016
 500 NEXT x
 525 REM Rysowanie osi Y
 600 POKE 24575,0: POKE 24574,255
 700 FOR y=0 TO 191
 800 POKE 24573,y
 900 RANDOMIZE USR 50016
1000 NEXT y
1025 REM Rysowanie funkcji dla zmiennej x=-s do s.
1100 LET k=0
1200 INPUT "s=";s,"e$=";e$
1250 PRINT "-s<=x<=s         y=";e$;""
1275 PRINT "s=";s
1300 FOR f=0 TO 511
1400 LET x=(f-256)*s/256: IF x=0 THEN  LET k=0: NEXT f
1500 LET y=96-INT (VAL e$)
1600 IF y<0 THEN  LET y=0: NEXT f
1700 IF y>191 THEN  LET y=191: NEXT f
1800 IF k=0 THEN  LET fs=f: LET ys=y: LET k=1
1900 LET dy=y-ys
2000 IF ABS dy<=1 THEN  GO TO 2400
2100 FOR i=SGN dy TO dy STEP SGN dy
2200 LET y=ys+i
2300 LET f=INT (fs+i/dy+.5)
2400 POKE 24573,y
2500 POKE 24575,INT (f/256): POKE 24574,f-256*INT (f/256)
2600 RANDOMIZE USR 50016
2700 IF ABS dy>1 THEN  NEXT i
2800 LET fs=f: LET ys=y
2900 NEXT f
3000 BEEP .5,12
3100 PAUSE 0
3200 STOP
3250 REM Kasowanie pamieci ekranow
3300 RANDOMIZE USR 50001: CLS
```

dodać do obliczonego adresu 8192. Adres komórki, w której leży dany punkt o współrzędnych X,Y może się więc składać z dwóch części:

— części a, obliczonej z zależności:

a = 16384+32*(INT((191−y)/8)−INT((191−y)/64)*8+8*(191−y−INT((191−y)/8)*8)+ +64*INT((191−y)/64))+INT(x/16)

— oraz części c, gdzie:

c = 8192*(INT(x/8)/2 > INT(INT(x/8)/2))

Dla kolumn parzystych warunek w nawiasie (= 0) i stąd c = 0. Dla kolumn nieparzystych warunek jest spełniony (= 1) i stąd c = 8192.

Ogólnie: adres = a+c

Pozostaje jeszcze obliczenie położenia bitu w bajcie. Zazwyczaj bity numerowane są od strony prawej do lewej, od 0 do 7. Natomiast po odjęciu od współrzędnej x wartości INT(x/8)*8 otrzymujemy numer bitu liczony od strony lewej do prawej, od 0 do 7. Przy wpisywaniu bitów do odpowiednich komórek pamięci wygodnie jest korzystać z tablicy. Ponieważ nie można stosować zmiennych o indeksie 0, zastosowano numerację bitów od strony lewej do prawej, od 1 do 8. Numer bitu w bajcie oznaczamy przez b i obliczamy z zależności:

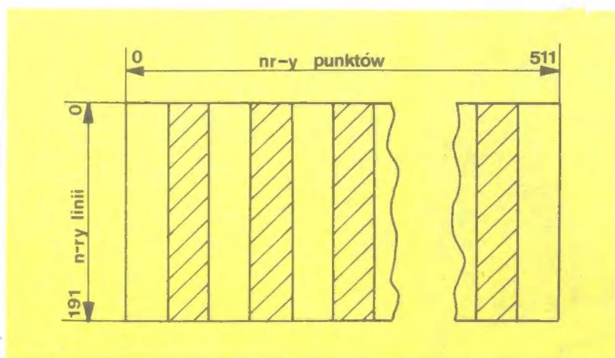
b = 1+x−INT(x/8)*8

Do wpisania bitu w odpowiednie miejsce w bajcie posłużono się tablicą jednowymiarową d(8), gdzie zmiennym d(1) do d(8) nadano wartości 128; 64; 32; 16; 8; 4; 2 i 1. Do komórki o adresie a+c wpisuje się wartość zmiennej d(b) przy pomocy rozkazu POKE sekwencją:

POKE a+c,d(b)

Pisząc program na rysowanie w trybie 4 np. wykresu funkcji należy koniecznie wprowadzić zabezpieczenie, żeby nie wpisywać punktów leżących poza obszarem pamięci ekranu, gdyż może to prowadzić do zniszczenia programu. Pozostaje jeszcze problem skasowania ekranu. Kolumny parzyste kasuje się instrukcją CLS. Natomiast, aby skasować kolumny nieparzyste należy do komórek pamięci drugiego ekranu wpisać zera.

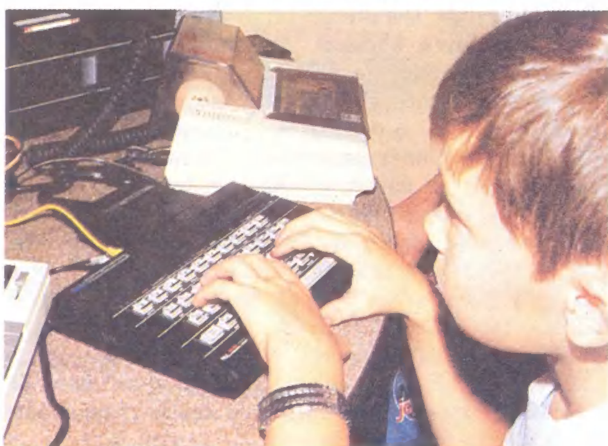




Rys. 3. Obraz w grafice 512×192 punkty (zakreskowano kolumny nieparzyste)

W trybie tym korzystanie z możliwości graficznych TIMEX-a jest najłatwiejsze, można bowiem na ekranie wyświetlać, redagować i poprawiać program oraz odczytywać wyświetlane komunikaty. Nie ma natomiast prążków przy wczytywaniu lub zapisywaniu programu na taśmie magnetofonowej.

Poniżej przedstawiono dwa programy w BASIC-u umożliwiające korzystanie z grafiki w trybie 4. Najpierw należy wpisać program **LD4.1**, zapisać go na taśmie, a następnie uruchomić komendą RUN. Programu tego nie można już później wylistować, gdyż początek programu w BASIC-u przesunął się do adresu 31489. Teraz można wpisać program **RF4.1**. Po wykonaniu komendy OUT 255,6 i uruchomieniu programu rysuje on osie współrzędnych o początku w środku ekranu. Można za jego pomocą rysować wiele funkcji, jak wielomiany i funkcje typu 1/x czy sin x/x. Jako s należy wstawić dowolną liczbę w zależności od żądanego zakresu zmiennej x, a jako e$ funkcję zmiennej x. Ponieważ na osi Y maksymalne wartości wynoszą ±96, należy niekiedy obliczone wartości funkcji pomnożyć przez odpowiednio dobrany współczynnik. Proponujemy jako przykład wstawić s = 5*PI oraz e$ = 70*sin x/x. Po zakończeniu rysowania funkcji wyświetlony zostanie komunikat STOP. Można nie kasować ekranu, tylko przejść do rysowania następnej funkcji przez GOTO 140 i ENTER lub skasować ekran przyciskając CONT i ENTER.



Program ten napisany całkowicie w BASIC-u działa bardzo wolno i posiada wiele innych wad. Na przykład nie łączy obliczonych punktów krzywej linią ciągłą. Udoskonalenie grafiki linii poziomych (linie 280 i 290) też nie jest całkowicie poprawne. Warto się jednak z nim zapoznać i ewentualnie wzory obliczania adresów punktów wykorzystać przy samodzielnym programowaniu. Znacznie szybszy i bardziej udoskonalony jest zestaw następnych dwóch programów: **LD4.2** i **RF4.2**. W programach tych obliczanie adresu punktu i kasowanie kolumn nieparzystych wykonywane jest w kodzie maszynowym, co znacznie przyspiesza działanie programu. Ponadto udoskonalono program rysowania wykresu funkcji wprowadzając łączenie obliczonych punktów krzywej pionowymi odcinkami. Natomiast jeśli w bajcie, do którego wstawia się bit, są już inne bity, to program w kodzie maszynowym dopisuje tylko nowe bity do już istniejących. Umożliwia to rysowanie linii poziomych i linii przecinających się.

Z programów tych korzysta się analogicznie jak z programów poprzednich. Najpierw wpisuje się program LD4.2. Przesuwa on RAMTOP do adresu 50000, a następnie od adresu 50001 wpisuje program w kodzie maszynowym oraz przesuwa początek programu w BASIC-u za obszar atrybutów drugiego ekranu, tj. do adresu 31489. Po wpisaniu programu LD4.2 nie należy go uruchamiać lecz najpierw zapisać na taśmie. Następnie uruchamiamy go i wpisujemy program RF4.2. Teraz można wykonać komendę OUT 255,6.

Po zapoznaniu się z obydwoma programami można je zmodyfikować tak, aby wpisywały się i uruchamiały same. W programie rysowania wykresów funkcji zrezygnowano z opisu osi, obliczania ekstremów i tzw. autoskalowania osi Y, aby niepotrzebnie nie zaciemniać i tak już dość zawiłych problemów korzystania z grafiki TIMEX-a.

Na zakończenie kilka uwag dotyczących korzystania z programu RF4.2. Program kasowania drugiego ekranu zajmuje 15 bajtów od adresu 50001 i uruchamia się przez RANDOMIZE USR 50001. Program w kodzie maszynowym do obliczania adresu bajtu i wpisania do niego bitu o współrzędnych x,y zaczyna się od adresu 50016 i można go uruchomić przez RANDOMIZE USR 50016. W celu przekazania współrzędnych x,y do programu w kodzie maszynowym należy y umieścić w komórce o adresie 24573, natomiast x w komórce 24574 — młodszy bajt i w 24575 — starszy bajt. Można wrócić do programu rysowania funkcji bez kasowania osi i krzywej już narysowanej komendą GOTO 1100. Po zakończeniu działania program zatrzymuje się na linii 3100 (PAUSE 0). Dzięki temu unika się kasowania wykresu komunikatami wyświetlanymi w oknie systemowym. Po przyciśnięciu dowolnego klawisza wyświetlany jest komunikat STOP. W celu skasowania ekranu należy przycisnąć CONT i ENTER. Linie od 1900 do 2300 służą do łączenia linią ciągłą punktów rysowanej krzywej.

Mimo iż artykuł ten nie wyczerpuje wszystkich zagadnień związanych z grafiką TIMEX-a 2048 to jednak mamy nadzieję, że zamieszczone w nim uwagi i programy umożliwiają poznanie możliwości graficznych tego mikrokomputera.



TADEUSZ ZALESKI

Procedura SCRCP na ZX Spectrum

Procedura pozwala na kompresję ekranu w pamięci mikrokomputera ZX Spectrum z możliwością wielokrotnego wywoływania i zapisu na taśmie magnetycznej. Objętość procedury wynosi 224 B, zaś sam zbiór bajtów opisujących ekran ulega komasacji z 6912 B do 2000...4000 B w zależności od stopnia wypełnienia ekranu (oczywiście możliwa jest do pomyślenia konstrukcja takiego obrazu, który po kompresji będzie zawierać... więcej niż 6912 B, jest to jednak mało prawdopodobne).

Po wprowadzeniu programu ładującego (wydruk 1) należy go uruchomić (RUN). Komunikat „0 OK, 280 : 1" oznacza poprawne zakończenie pracy i pozwala na zapisanie procedury SCR CP na taśmie (SAVE „SCRCP" CODE 58368,224).

Kompresję ekranu (np. zapisanego uprzednio w pełnej formie na taśmie magnetofonowej) realizuje kolejny program (wydruk 2a). Oczywiście linia 30 może być zastąpiona procedurą generującą obraz na ekranie monitora. Użycie tak skondensowanej informacji o zawartości ekranu demonstruje wydruk 2b. Linia 30 może być wywoływana wielokrotnie, co powoduje ponowne odtworzenie zawartości ekranu. Można również zapamiętać nowy rysunek dodając linię 40 z wydruku 2a oraz zapisać go na taśmie magnetofonowej (linia 50 z wydruku 2a), gdyż wraz z rysunkiem na taśmie zapisujemy procedurę komasującą informację o ekranie w pamięci (SCRSV), jak i dekodującą tę informację z przesłaniem jej do obszaru D FILE (SCRLD).

Dla bardzo oszczędnych (chodzi o czas nagrania, miejsce w pamięci mikrokomputera i na taśmie) można zaproponować wariant „oszczędny" (wydruki 3a i 3b). Zapis będzie krótszy o 120 B, lecz nie ma możliwości zapamiętania nowego rysunku (na taśmie znajduje się jedynie procedura SCRLD).

Wydruk 4 przedstawia treść obu procedur w postaci symbolicznej asemblera Z80. Procedura SCRSV (SCREEN SAVE) odczytuje informację stanowiącą widoczny na ekranie obraz, kondensuje ją i umieszcza w obszarze pamięci począwszy od komórki o nazwie SCR, przy czym odmienny jest sposób kodowania informacji o samym obrazie (linie 60...470) i o jego atrybutach (linie 490...770). Dla obrazu zapisywana jest informacja o ilości kolejnych komórek zawierających 00H (linie 100...310), a następnie sekwencja zawartości następnych komórek różnych od 00H, po napotkaniu ponownie komórki o zawartości równej 00H (linie 440...470) jest ona zapisywana jako symbol końca ciągu różnego od 00H i ponownie zliczane są kolejne komórki zawierające 00H. Procedura zostanie przerwana w chwili, gdy sprawdzona zostanie łączna ilość 1800H (6144) komórek stanowiących pełną informację o treści ekranu (rolę licznika spełnia para rejestrów DE, linie 110...130 i 320...340). Dla atrybutów zliczana jest ilość kolejnych komórek o identycznej zawartości (linie 500...620), w chwili gdy wykryta zostanie komórka o innej zawartości niż analizowana (linie 650...740), to w pamięci zapisana zostaje informacja

o ilości komórek z identyczną zawartością (linie 660...690), jak i sama zawartość (linia 710). Cykl powtarzany jest tak długo, aż przeanalizowane zostanie 0300H (768) komórek (tu również rolę licznika pełni para rejestrów DE, linie 530...550 i 750...770). Analiza wielu obrazów zdaje się wskazywać, że zaproponowany sposób kondensacji jest niemal optymalny.

Z powyższego opisu kodowania sposób odtwarzania obrazu przez procedurę SCRLD (SCREEN LOAD) wydaje się być oczywisty i pozostawiam go bardziej wnikliwym Czytelnikom.

Wydruk 1

```
 10 CLEAR 58367: RESTORE
 20 LET ADD=58368
 30 FOR I=0 TO 13
 40 READ Z$,T
 50 LET S=0
 60 FOR J=0 TO 15
 70 LET XH=CODE Z$(1)-48: IF XH>9 THEN LET XH=XH-7
 80 LET XL=CODE Z$(2)-48: IF XL>9 THEN LET XL=XL-7
 90 LET Z$=Z$(3 TO )
100 POKE ADD+16*I+J,16*XH+XL
110 LET S=S+16*XH+XL
120 NEXT J
130 IF S<>T THEN PRINT "BLAD W DANYCH": STOP
140 NEXT I
150 DATA "187621DBE42276E42100401100180100",1141
160 DATA "007BB228097EB7200503231B18F3E52A",1299
170 DATA "76E4712370232276E44D44E17BB22812",1750
180 DATA "7EB728060203231B18F20203ED4376E4",1343
190 DATA "18CC1100037E010000F57BB22809F1BE",1401
200 DATA "200603231B18F2F1E52A76E471237023",1522
210 DATA "77232276E4E17BB220DB2A76E41100E4",1944
220 DATA "A7ED524D44C9000021DBE42276E42100",1725
230 DATA "40110018E52A76E44E2346232276E4E1",1545
240 DATA "7BB2282179B02807AF770B1B2318F1ED",1587
250 DATA "4B76E47BB2280E0A03ED4376E4B728D4",1874
260 DATA "771B2318EE110003E52A76E44E234623",1298
270 DATA "7E232276E4E1F57BB22002F1C979B028",2125
280 DATA "07F177231B0B18EEF118DD0061E1F57B",1878
```

Wydruk 2a

```
10 CLEAR 58367
20 LOAD "SCRCP"CODE
30 LOAD ""SCREEN$
40 POKE 22527,0: LET L=USR 58370
50 SAVE "nazwa"CODE 58368,L
```

Wydruk 2b

```
10 CLEAR 58367
20 LOAD "nazwa"CODE
30 RANDOMIZE USR 58368
```

Wydruk 3a

```
10 CLEAR 58367
20 LOAD "SCRCP"CODE
30 LOAD ""SCREEN$
40 POKE 22527,0: LET L=USR 58370
50 SAVE "nazwa"CODE 58488,L-120
```

Wydruk 3b

```
10 CLEAR 58485
20 LOAD "nazwa"CODE
30 RANDOMIZE USR 58488
```

Wydruk 4

```
0010            ORG    E400H
0020 ;          SCREEN_LOAD
0030 START      JR     SCRLD
0040 ;          SCREEN_SAVE
0050 ;          TAZ(C)1986
0060 SCRSV      LD     HL,SCR      ;ustaw warunki początkowe
0070            LD     (REG),HL
0080            LD     HL,4000H
0090            LD     DE,1800H
0100 SVBPO      LD     BC,0000H    ;zliczaj bajty obrazu = 0
0110 SVBNO      LD     A,E         ;pętla zliczająca
0120            OR     D
0130            JR     Z,SVBDL     ;skocz, gdy wszystkie bajty
0140            LD     A,(HL)
0150            OR     A
0160            JR     NZ,SVBDL    ;skocz, gdy bajt = 0
0170            INC    BC
0180            INC    HL
0190            DEC    DE
0200            JR     SVBNO
0210
0220 SVBDL      PUSH   HL          ;zapamiętaj ilość bajtów = 0
0230            LD     HL,(REG)
0240            LD     (HL),C
0250            INC    HL
0260            LD     (HL),B
0270            INC    HL
0280            LD     (REG),HL
0290            LD     C,L
0300            LD     B,H
0310            POP    HL
0320 SVBN1      LD     A,E         ;zapamiętaj w pętli bajty <> 0
0330            OR     D
0340            JR     Z,SVAST     ;skocz, gdy wszystkie bajty
0350            LD     A,(HL)
0360            OR     A
0370            JR     Z,SVBK1     ;skocz, gdy bajt = 0
0380            LD     (BC),A
0390            INC    BC
0400            INC    HL
0410            DEC    DE
0420            JR     SVBN1
0430 ;
0440 SVBK1      LD     (BC),A      ;zaznacz koniec bajtów <> 0
0450            INC    BC
0460            LD     (REG),BC
0470            JR     SVBPO
0480 ;
0490 SVAST      LD     DE,0300H    ;analizuj bajty atrybutów
0500 SVANA      LD     A,(HL)      ;pętla analizująca
0510            LD     BC,0000H
0520 SVANX      PUSH   AF          ;pętla zliczająca bajty = A
0530            LD     A,E
0540            OR     D
0550            JR     Z,SVAKO     ;skocz, gdy wszystkie bajty
0560            POP    AF
0570            CP     (HL)
0580            JR     NZ,SVADL    ;skocz, gdy bajt <> od A
0590            INC    BC
0600            INC    HL
0610            DEC    DE
0620            JR     SVANX
0630 ;
0640 SVAKO      POP    AF          ;zakończenie analizy
0650 SVADL      PUSH   HL          ;zakończenie dla danego A
0660            LD     HL,(REG)
0670            LD     (HL),C
0680            INC    HL
0690            LD     (HL),B
0700            INC    HL
0710            LD     (HL),A
0720            INC    HL
0730            LD     (REG),HL
0740            POP    HL
```

```
0750            LD     A,E
0760            OR     D
0770            JR     NZ,SVANA ;skocz, gdy analizować dalej
0780            LD     HL,(REG)
0790            LD     DE,START
0800            AND    A
0810            SBC    HL,DE
0820            LD     C,L
0830            LD     B,H
0840            RET             ;w BC łączna długość zapisu
0850 ;
0860 REG        DEFW   0000H    ;rejestr pomocniczy
0870 ;          SCREEN_LOAD
0880 ;          TAZ(C)1986
0890 SCRLD      LD     HL,SCR   ;ustaw warunki początkowe
0900            LD     (REG),HL
0910            LD     HL,4000H
0920            LD     DE,1800H
0930 LDBDL      PUSH   HL       ;wpisuj bajty = 0
0940            LD     HL,(REG)
0950            LD     C,(HL)
0960            INC    HL
0970            LD     B,(HL)
0980            INC    HL
0990            LD     (REG),HL
1000            POP    HL
1010 LDBN0      LD     A,E      ;pętla wpisująca bajty = 0
1020            OR     D
1030            JR     Z,LDAST  ;skocz, gdy zapełniono ekran
1040            LD     A,C
1050            OR     B
1060            JR     Z,LDBK0  ;skocz, gdy to wszystkie bajty
1070            XOR    A
1080            LD     (HL),A
1090            DEC    BC
1100            DEC    DE
1110            INC    HL
1120            JR     LDBN0
1130 ;
1140 LDBK0      LD     BC,(REG) ;wpisuj bajty <> 0
1150 LDBN1      LD     A,E      ;pętla dla bajtów <> 0
1160            OR     D
1170            JR     Z,LDAST  ;skocz, gdy zapełniono ekran
1180            LD     A,(BC)
1190            INC    BC
1200            LD     (REG),BC
1210            OR     A
1220            JR     Z,LDBDL  ;skocz, gdy bajt = 0
1230            LD     (HL),A
1240            DEC    DE
1250            INC    HL
1260            JR     LDBN1
1270 ;
1280 LDAST      LD     DE,0300H ;wpisuj bajty atrybutów
1290 LDADL      PUSH   HL       ;pętla główna
1300            LD     HL,(REG)
1310            LD     C,(HL)
1320            INC    HL
1330            LD     B,(HL)
1340            INC    HL
1350            LD     A,(HL)
1360            INC    HL
1370            LD     (REG),HL
1380            POP    HL
1390 LDANX      PUSH   AF       ;petla wpisująca atrybuty
1400            LD     A,E
1410            OR     D
1420            JR     NZ,LDACT ;skocz, gdy jeszcze wpisywać
1430            POP    AF
1440            RET
1450 ;
1460 LDACT      LD     A,C      ;kontynuacja pętli wpisującej
1470            OR     B
1480            JR     Z,LDANA  ;skocz, gdy to wszystkie bajty
1490            POP    AF
1500            LD     (HL),A
1510            INC    HL
1520            DEC    DE
1530            DEC    BC
1540            JR     LDANX
1550 ;
1560 LDANA      POP    AF       ;kontynuacja pętli głównej
1570            JR     LDADL
1580 ;
1590 SCR        DEFB   00H      ;obszar, gdzie zapisywana jest
1600 ;                           skompensowana informacja o
1610 ;                           ekranie
1620            END
```

Ciąg dalszy ze s. 11

szczyzny (obroty i przesunięcia). Autorzy sygnalizują również problemy animacji na płaszczyźnie. Następna część to już wyższa szkoła programowania: grafika trójwymiarowa. Standardem jest już rysowanie wykresu funkcji dwóch zmiennych na ekranie monitora. Autorzy jednak na tym nie poprzestają i przedstawiają sposoby wykonywania rzutów równoległych i perspektywicznych brył na płaszczyznę, a także złożone operacje przekształcania przestrzeni trójwymiarowej. Książka kończy się omówieniem metod graficznego przedstawiania danych np. w postaci diagramów słupkowych (ang. *bar charts*) i krążkowych (ang. *pie charts*).

Każde zagadnienie zilustrowane jest programem napisanym w BASIC-u. Wszystkie napisane są w sposób niezwykle przejrzysty z załączonymi obszernymi komentarzami po angielsku (to cenna uwaga dla tych, którzy mają kłopoty z językiem rosyjskim). Jedynymi



instrukcjami graficznymi są instrukcje PLOT i DRAW, co pozwala na wpisywanie programów przez posiadaczy sprzętu każdego rodzaju.

Książka została napisana w 1983 r., kiedy to zewnętrzne urządzenia graficzne dla komputerów domowych nie były rozwinięte w takim stopniu jak obecnie. Nie ma np. informacji o myszy czy piórze świetlnym. Mimo takich, drobnych zresztą, niedostatków książka „Mikrokomputernaja grafika" powinna stanowić cenną pomoc dla wszystkich miłośników informatyki, a szczególnie dla tych, którzy lubią także matematykę. Bez niej w grafice komputerowej można zdziałać naprawdę niewiele.

Operacjonnaja systiema MS-DOS, Moskwa 1987; tytuł oryginału: Making PC-DOS & MS-DOS Work For You, rok wydania 1984. Za drobną sumę 60 zł można było nabyć doskonały podręcznik najpopularniejszego systemu operacyjnego komputerów standardu IBM PC. W większości książka jest wiernym tłumaczeniem amerykańskiego oryginału i jak gros podręczników zza oceanu rozpoczyna się od przystępnego wyjaśnienia po co komputerowi w ogóle jest potrzebny system operacyjny, co to jest MS-DOS i jak poczynić z nim pierwsze kroki. W książce nie pojawia się następnie bezmyślny spis wszystkich zleceń: podręcznik pogrupowany jest na rozdziały wprowadzające w tajniki systemu operacyjnego stopniowo. I tak zaczyna się od spisu zleceń do manipulowania plikami dyskowymi tj. Copy, Dir, Rename, Del itp. Następnie jest mowa o uruchamianiu programów w systemie MS-DOS, o dyskietkach, o przełączaniu logicznych numerów stacji dysków

Dalszy ciąg na s. 15

MACIEJ GONET

Jak przechowywać wyniki obliczeń w ZX Spectrum ?

Każdy z Czytelników, kto próbował wykonywać przy użyciu ZX Spectrum obliczenia naukowe, techniczne czy ekonomiczne, odczuł zapewne potrzebę utrwalenia wyników, szczególnie gdy ich objętość jest duża, a obliczenia — długotrwałe. Można oczywiście wyprowadzać wszystko na drukarkę (jeśli się ją posiada), ale takie rozwiązanie — choć najprostsze — nie zawsze jest najwłaściwsze. Dużo wygodniej przechowywać wyniki obliczeń w pamięci operacyjnej komputera lub zapisać je na zewnętrznym nośniku magnetycznym, aby mieć możliwość ich przeglądnięcia i analizy przed ewentualnym wydrukowaniem.

Realizacja tego celu na poziomie BASIC-a jest możliwa, choć w instrukcji dostarczanej przez producenta nie została ona dostatecznie opisana. Należy tylko zamiast wyprowadzania wyników za pomocą instrukcji PRINT (lub równolegle) utworzyć łańcuch zawierający wszystkie wyprowadzane informacje. BASIC ZX Spectrum umożliwia bowiem zamianę wszystkiego, co można umieścić w instrukcji PRINT na odpowiednie łańcuchy lub znaki sterujące. Np. zamiast instrukcji

```
100 PRINT AT 5,2;"WYNIK:",pole;
" m^2"
```

piszemy

```
100 LET w$=CHR$ 22+CHR$ 5+CHR$
2+"WYNIK:"+CHR$ 6+STR$ pole+" m^
2"+CHR$ 13
```

W zamieszczonej tabeli zestawiono wszystkie możliwe elementy instrukcji PRINT i ich odpowiedniki w łańcuchu.

Instrukcja PRINT	Łańcuch
PRINT...	LET w$ = w$+...+CHR$ 13
125	„125"
x	STR$ x
„tekst"	„tekst"
a$	a$
; (średnik)	+
, (przecinek)	CHR$ 6
' (apostrof)	CHR$ 13
INK n	CHR$ 16+CHR$ n
PAPER n	CHR$ 17+CHR$ n
FLASH n	CHR$ 18+CHR$ n
BRIGHT n	CHR$ 19+CHR$ n
INVERSE n	CHR$ 20+CHR$ n
OVER n	CHR$ 21+CHR$ n
AT m,n	CHR$ 22+CHR$ m+CHR$ n
TAB n	CHR$ 23+CHR$ n+CHR$ 0

Należy pamiętać, że instrukcja PRINT nie zakończona separatorem w postaci średnika lub przecinka przesyła na końcu znak zmiany wiersza CHR$ 13, który należy uwzględnić w tworzonym łańcuchu. Pewna trudność może pojawić się na skutek jednego z błędów zawartych w pamięci ROM ZX Spectrum, określanego czasem mianem „żarłocznej" funkcji STR$. Polega on na nieprawidłowym obliczaniu wartości wyrażenia typu

w$+STR$ x   gdzie $|x| < 1$.

Łańcuch w$ po lewej stronie funkcji STR$ jest pomijany (zastępowany łańcuchem pustym). Przewidując pojawienie się w wynikach liczb ułamkowych z zakresu −1 do 1 należy zamienić liczbę na łańcuch w oddzielnej instrukcji

```
LET b$ = STR$ x
```

i dopiero potem dołączyć do łańcucha wynikowego

```
LET w$ = w$+b$
```

Po zakończeniu obliczeń łańcuch w$ można wyświetlić na ekranie lub wydrukować. Można go także zapisać na taśmie magnetofonowej zleceniem SAVE w postaci:

```
SAVE "wynik" DATA w$()
```

i ewentualnie zweryfikować zapis zleceniem VERIFY w podobnej postaci. Kłopoty pojawią się jednak po wczytaniu łańcucha instrukcją LOAD. Autorzy programu zawartego w ROM-ie nie przewidywali widocznie zapisywania łańcuchów prostych, ale też nie zapobiegli dokonywaniu takiego zapisu. W efekcie łańcuch zapisany na taśmie można wczytać do pamięci, ale nie można go odczytać za pomocą instrukcji BASIC-a.

Istnieje kilka możliwości pokonania tego problemu. Jednym z nich jest przepisanie łańcucha do tablicy i zapisanie go na taśmie w tej formie np.

```
DIM y$(LEN w$): LET y$=w$: SAVE
"wynik" DATA y$()
```

Po wczytaniu można ewentualnie przepisać tablicę z powrotem do zmiennej łańcuchowej.

Innym sposobem jest „poprawienie" błędu pamięci ROM za pomocą krótkiego programu w BASIC-u:

```
10 LET ad=0
20 LET ad=PEEK 23641+256*PEEK
23642-1
30 LOAD "wynik" DATA w$()
40 POKE ad,PEEK ad-128
50 PRINT w$
```

Działanie tego programu jest następujące: w linii 10 rezerwujemy miejsce w obszarze zmiennych, następnie (linia 20) wyznaczamy adres końca obszaru zmiennych (E-LINE) — 1. Po wczytaniu łańcucha w$ (linia 30) jego nazwa zostanie umieszczona w pamięci pod tym właśnie adresem. Pozostaje teraz wyzerować siódmy bit owego bajtu (linia 40), aby przywrócić właściwą formę zapisu łańcucha, który teraz może być już bez przeszkód odczytany (linia 50).

Na zakończenie trzeba uczciwie przyznać, że proponowany pośredni sposób wyprowadzania wyników przedłuża wyraźnie czas wykonania programu. Nie polecam go więc do programów, w których jest niewiele obliczeń. Warto go natomiast stosować, gdy program wyprowadza co kilka minut porcję wyników o objętości ekranu.

Dysk pamięciowy w ZX Spectrum

DARIUSZ ADAM PRZYGODA
KRZYSZTOF AMBORSKI

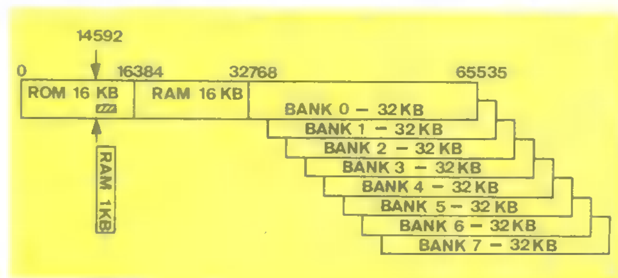
Użytkownicy ZX Spectrum stosujący ten mikrokomputer do pracy w językach ściśle związanych ze środowiskiem systemowym (asembler, C) zapewne wielokrotnie mieli okazję zaobserwować reakcję komputera na popełnione błędy programowe polegającą na „zawieszeniu" się systemu, wymagającym użycia przycisku RESET, albo (częściej) samoczynnej reinicjalizacji komputera. W obu przypadkach Spectrum testuje swoją pamięć zapełniając jej komórki wartościami zero, co powoduje całkowitą utratę poprzedniej zawartości.

Posiadacze przystawki ZX Interface-1/Microdrive lub stacji dysków mogą szybko przywrócić stan poprzedni, jednak w wypadku magnetofonu ładowanie programu trwa znacznie dłużej (średnio 1 KB/5,2 s), do czego dochodzą jeszcze niewygodne manipulacje kasetami. Chcielibyśmy zaproponować Czytelnikom pewne rozwiązanie sprzętowe pozwalające uniknąć takich nieprzyjemnych sytuacji. Jest to dysk pamięciowy (ang. *RAM Disc*) czyli wydzielony obszar pamięci o dostępie swobodnym RAM pełniącej rolę analogiczną do zewnętrznej pamięci masowej. Informacja zapamiętana w obszarze dysku pamięciowego nie podlega utracie podczas inicjalizacji systemu i może być szybko załadowana do obszaru roboczego komputera. Wadą urządzenia jest utrata informacji w momencie odłączenia komputera od sieci zasilającej. Gospodarką zasobami dysku pamięciowego zajmuje się specjalny program obsługi.

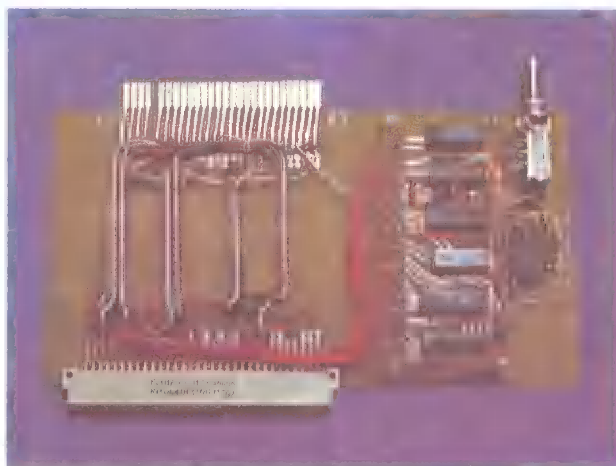
Proponowane poniżej rozwiązanie charakteryzuje się następującymi parametrami technicznymi:
- pojemność dysku pamięciowego 224 KB,
- prędkość transmisji ok. 15 KB/s (zależy od rodzaju zastosowanych pamięci; podany przypadek najmniej korzystny),
- dostęp do zasobów dysku z poziomu języka BASIC dodatkowymi instrukcjami (własna obsługa błędów).

Struktura logiczna dysku pamięciowego przedstawiona jest na rys. 1. Dysk pamięciowy tworzą obszary wymienialne z najwyższymi 32 KB pamięci komputera (zwane dalej umownie „bankami"). Wymiana banków dokonuje się z poziomu programu przez wpisanie do portu o adresie #FF (255) numeru banku (0—7). Program obsługi dysku pamięciowego umieszczony jest w dodatkowym obszarze pamięci RAM nakładkowanym na nie używany obszar pamięci stałej ROM komputera.

Rys. 1. Schemat logiczny obszarów pamięci komputera



Proponowane rozwiązanie stanowi znaczne rozszerzenie możliwości komputera ZX Spectrum. Charakteryzuje się ono jednak dość znaczną ceną (szacowany w III kwartale 1987 roku koszt przeróbki wynosił ok. 30 000 zł, co stanowiło mniej więcej 25% ceny giełdowej komputera ZX Spectrum), a także koniecznością znacznej ingerencji w konstrukcję komputera. Dlatego też odradzamy podejmowanie prób jego budowy początkującym adeptom elektroniki komputerowej. Ewentualny błąd skończyć się może uszkodzeniem zarówno komputera, jak i drogich elementów dysku pamięciowego. Dodać też należy, że uruchomienie urządzenia wymaga pewnej praktyki zarówno sprzętowej, jak i programistycznej.

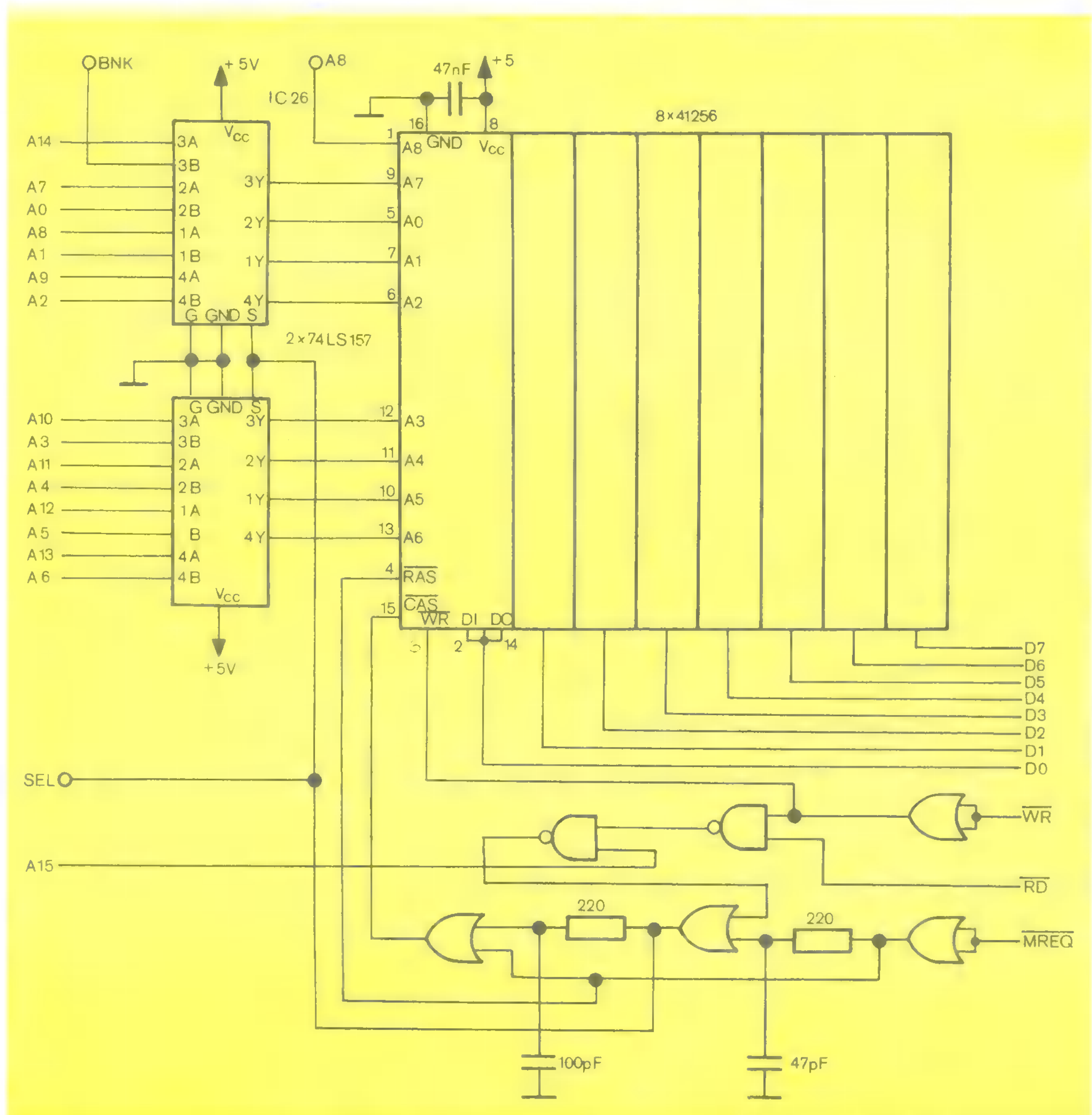


Płytka układu rozszerzenia pamięci do 256 KB

W pierwszej części artykułu przedstawimy opis konstrukcyjny dysku pamięciowego, zaś w drugiej — listing i opis programu jego obsługi.

Opis części sprzętowej dysku pamięciowego

Schemat dysku pamięciowego ze względów funkcjonalnych został podzielony na następujące bloki:
1. Blok pamięci 256 KB (jest to w zasadzie już istniejący blok pamięci 32 KB z wprowadzonymi modyfikacjami).



Rys. 2. Schemat ideowy bloków pamięci 256 KB. Uwaga! Wyprowadzenia 10 i 11 układu scalonego IC 26 (74LS157) należy odłączyć rozcinając zwory J1 i J2 i połączyć zgodnie z rysunkiem. Wyprowadzenia 1 pamięci (A8) połączyć przewodem

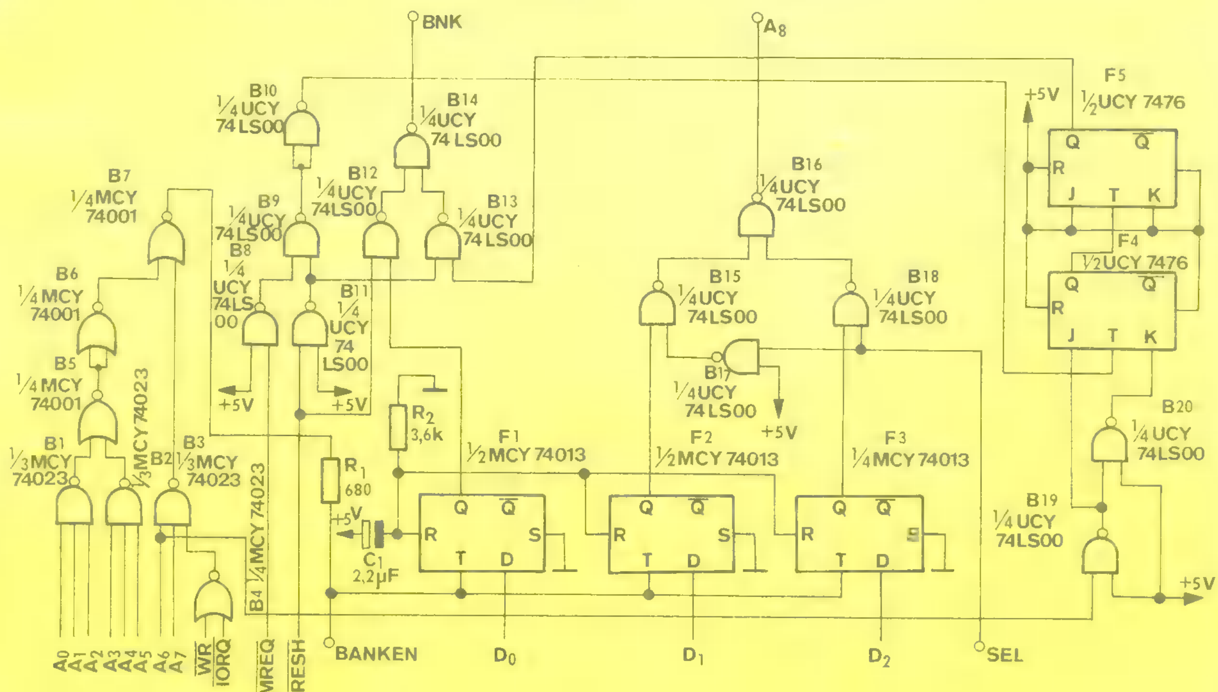
2. Blok dodatkowych układów sterujących pamięcią 256 KB.
3. Rozszerzenie pamięci o dodatkowy 1 KB pamięci o dostępie swobodnym RAM.
4. Zmodyfikowany układ sterujący sygnałem RESET.

Poniżej podane zostały krótkie opisy poszczególnych bloków wyjaśniające ich działanie.

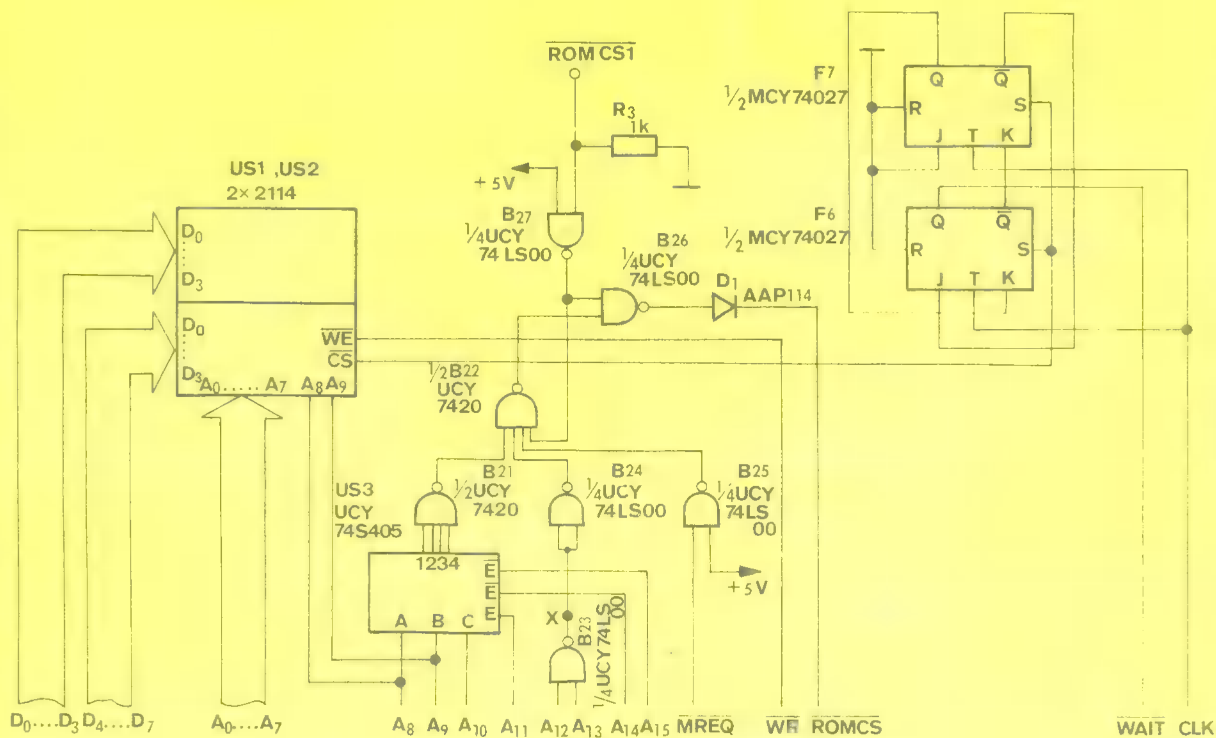
Blok pamięci 256 KB RAM

Jak wynika z rys. 2 schemat ideowy bloku niewiele różni się od schematu analogicznego bloku ZX Spectrum (schemat komputera publikowany był m.in. w miesięczniku „Informatyka", nr 6/85). Różnice sprowadzają się do odmiennego sterowania linii A14 (w ZX Spectrum 48 K[illegible] linie te podawana jest wartość stała 1 lub 0 w zależności od typu zastosowanych pamięci 32 KB) i dodatkowego sterowania (nie istniejącej w ZX Spectrum 48 KB linii A8). Dzięki zgodności wyprowadzeń układów pamięci 4532, 4164 i 41256 zmiany montażowe są niewielkie.

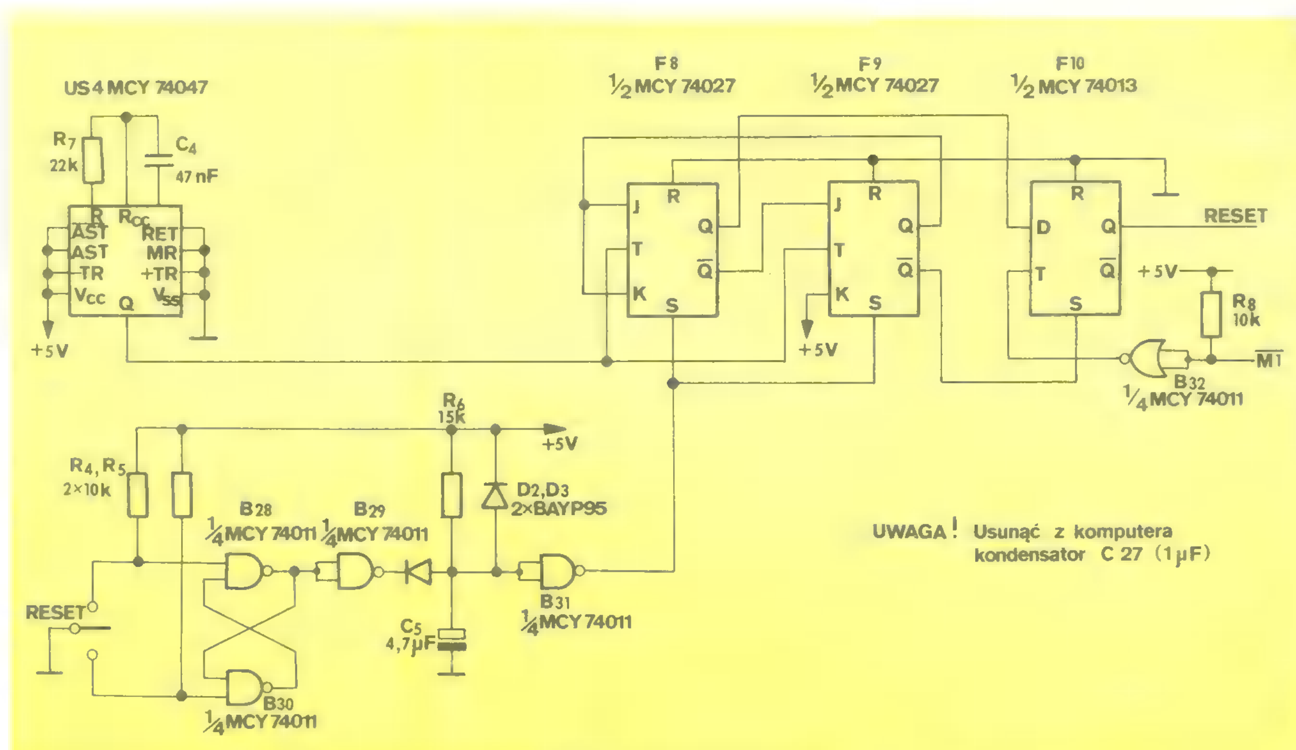
Aby wyjaśnić działanie układu niezbędne jest krótkie przypomnienie sposobu adresacji pamięci dynamicznych w obudowach 16-nóżkowych. Ze względu na brak odpowiedniej ilości wyprowadzeń adres do tych pamięci wprowadza się dwuczęściowo: najpierw niższą połówkę słowa adresowego, a następnie wyższą. O tym jak mają być interpretowane sygnały na szynach adresowych decydują sygnały /RAS i /CAS. W ZX Spectrum do generacji tych sygnałów użyto bramek TTL i układów



Rys. 3. Schemat ideowy dodatkowych układów sterujących pamięcią 256 KB



Rys. 4. Schemat ideowy układu rozszerzenia pamięci o dodatkowy 1 KB

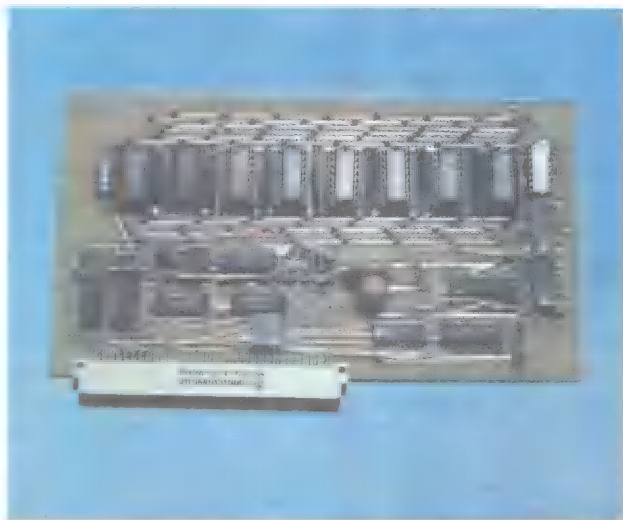


Rys. 5. Schemat ideowy zmodyfikowanego układu RESET

RC (widoczne na rys. 2). Dystrybucja słowa adresowego zrealizowana jest za pomocą multiplekserów 74157.

Pamięci 4532 to w rzeczywistości odrzuty produkcyjne pamięci 4164 z jedną połówką uszkodzoną; w zależności od fizycznej lokalizacji defektu wymagają one podania na szynę A15 jedynki lub zera logicznego w celu uaktywnienia dobrej połówki (w ZX Spectrum realizacja zworą J1). Dekoder skonstruowany jest w ten sposób, że lokalizacja adresowa obszaru pamięci (od adresu #8000) jest wymuszona sprzętowo. Rozwiązanie takie podyktowane zostało wykorzystaniem pamięci 4532; dla nas jest dużym ułatwieniem przy konstrukcji dysku pamięciowego.

Płytka pamięci przechowującej program obsługi dysku pamięciowego



Z powyższego wynika, że system adresacji dysku nie różni się w zasadzie od systemu adresacji ZX Spectrum. O ile jednak w komputerze wybór „dobrej połówki" pamięci 4532 realizowany jest zworą, o tyle zastosowanie układów 41256 daje nam możliwość wyboru jednej z ośmiu części (każda o pojemności 32 KB), a jest to realizowane elektronicznie. Służą do tego szyny BNK (dawny wybór zworą) i A8 (nowe wyprowadzenie realizujące sterowanie 17- i 18-tym bitem adresu, oczywiście w trybie dwuczęściowym, jak wszystkie wyprowadzenia adresowe pamięci). Podanie na te szyny jednej z ośmiu możliwych kombinacji powoduje wybór banku pamięci na tej samej zasadzie jak poprzednio wybierana była „połówka" pamięci. Do celów sterujących wyprowadzono także sygnał SEL służący do sterowania multiplekserami.

Blok dodatkowych układów sterujących pamięcią 256 KB

Aby opisany powyżej blok pamięci 256 KB mógł poprawnie funkcjonować konieczne jest wprowadzenie układu zapewniającego generację odpowiednich sygnałów sterujących. Blok ten powinien zapewnić podawanie zadanych sygnałów wyboru banków na odpowiednie szyny adresowe pamięci oraz prawidłowe odświeżanie zawartości pamięci. Schemat układu realizującego powyższe funkcje przedstawiony został na rys. 3.

Realizacja pierwszego wymagania nie wymaga komentarza. Bieżący bank wybierany jest zawartością trzybitowego rejestru (przerzutniki F1, F2 i F3), który widziany jest przez komputer jako trzy najmłodsze bity portu o adresie #FF (255). Dostęp do portu (zapis) zapewnia dekoder zbudowany przy użyciu bramek B1—B7. Wyprowadzona szyna BANKEN umożliwia sprzętowe zablokowanie dostępu do mechani-

zmu zmiany banków. Elementy R2 i C2 służą do zapewnienia ustawienia banku 0 jako bieżącego po włączeniu systemu. Bramki B15—B18 pełnią rolę multipleksera do przełączania 16. i 17. szyny adresowej. Krótkiego wyjaśnienia wymaga realizacja drugiego wymagania tj. prawidłowego odświeżania zawartości pamięci. Jak wiadomo informacja w pamięciach dynamicznych fizycznie zapamiętywana jest w postaci ładunku na kondensatorze. Najczęściej jest to ładunek na pojemności bramka-źródło (G-S) tranzystora MOS. Ze względu na zjawisko upływności odpowiadający zapamiętanej informacji ładunek elektryczny co pewien czas musi być odświeżany, gdyż inaczej nastąpi jego zanik. Producenci pamięci gwarantują czas przechowywania informacji bez zaniku wynoszący 2 ms.

Konstrukcja struktury układu pamięci powoduje, że proces odświeżania w przypadku użycia pamięci typu 4164 (i 4532) sprowadza się do konieczności zaadresowania 128 kolejnych lokacji (odpowiadających numerom rzędów matrycy struktury układu) w przeciągu 2 ms. W przypadku użycia pamięci typu 41256 ilość tych lokacji jest dwukrotnie większa.

Mikroprocesor Z80 wbudowane ma mechanizmy służące odświeżaniu pamięci w sposób „przezroczysty" dla systemu, jednakże generuje tylko 128 adresów (siedmiobitowe słowo odświeżania). Zachodzi więc konieczność wygenerowania pozostałych 128 lokacji. Problem ten sprowadza się w praktyce do wytworzenia ósmego bitu słowa odświeżającego.

Ramy niniejszego artykułu uniemożliwiają dokładniejsze omówienie tego problemu; dociekliwy Czytelnik może je znaleźć w publikacji pt. „Przystosowanie mikroprocesora Z80 do współpracy z pamięciami dynamicznymi o dużej pojemności", która ukaże się w jednym z najbliższych numerów „Wiadomości Elektrotechnicznych".

Do generacji ósmego bitu słowa odświeżania wykorzystano zmiany (generowanego przez procesor) siódmego bitu słowa odświeżania. Jest on zapamiętywany w przerzutniku F4. Przerzutnik F5 jest dzielnikiem przez dwa — na jego wyjściu Q uzyskiwany jest przebieg ósmego bitu odświeżania. Bramki B12—B14 realizują dalszą jego dystrybucję.

Blok układu rozszerzenia pamięci o dodatkowy 1 KB

Dodatkowa pamięć RAM o pojemności 1 KB służy do przechowywania programu obsługi dysku pamięciowego. Jest ona umieszczona w obszarze pamięci stałej ROM zawierającej system od adresu #3900. Lokalizację taką umożliwia fakt niewykorzystania przez programistów Sinclair Research Ltd. całej przestrzeni pamięci stałej — pozostało w niej ponad 1 KB wolnego miejsca zapełnionego wartością #FF.

Schemat układu przedstawiony jest na rys. 4. Użyte zostały elementy typu 2114 — pamięć statyczna o organizacji 4×1 KB (układy US1 i US2), ze względu na fakt posiadania przez nie wbudowanych buforów (łatwość dołączania układów do szyny). Dekoder adresu zawierający bramki B21—B27 i układ US3 powoduje — po jego uaktywnieniu adresem zawartym pomiędzy #3900 a #3CFF — odłączenie systemowej pamięci stałej i uaktywnienie dodatkowej pamięci RAM. Wejście /ROMCS1 jest zewnętrznym odpowiednikiem wejścia /ROMCS szyny ZX Spectrum.

Ze względu na różny czas dostępu pamięci 2114 (w zależności od producenta) wprowadzony został układ spowalniający (przerzutniki F6 i F7) umożliwiający współpracę mikroprocesora z elementami o dłuższym czasie dostępu. Przy wystarczająco szybkich elementach układ ten można pominąć.

Dla odłączenia dysku pamięciowego mikrokomputera zaleca się wyłączenie dekodera przez rozwarcie przełącznikiem jednej z linii sterujących oznaczonej na rys. 4 literą x.

Blok zmodyfikowanego układu RESET

Zmiany fabrycznego układu RESET komputera wymuszone zostały następującymi własnościami układowymi mikroprocesora Z80:

1. Na czas trwania sygnału RESET mikroprocesor zatrzymuje proces odświeżania pamięci dynamicznych. Wynika z tego, że przy czasie trwania tego sygnału dłuższym niż 2 ms może nastąpić utrata zawartości pamięci dynamicznych.
2. Wada konstrukcyjna mikroprocesora powoduje, że jeżeli sygnał RESET pojawi się podczas trwania taktu T2 lub T4 może nastąpić niekontrolowana zmiana sygnału /MREQ powodująca uszkodzenie zawartości pamięci.

W komputerze ZX Spectrum bez dysku pamięciowego oba te zjawiska są nieistotne, gdyż przy inicjalizacji systemu testowana jest cała pamięć, co powoduje utratę jej poprzedniej zawartości. Schemat układu RESET przedstawiony jest na rys. 5. Generator astabilny zbudowany przy użyciu elementu US4 generuje falę prostokątną, którą synchronizowany jest licznik zbudowany na elementach F8 i F9. Licznik ten został zaprojektowany tak, aby jego cykl zliczania sprowadził się do generacji impulsu równego dwóm okresom przebiegu generatora. Układ ten okazał się najkorzystniejszy pod względem odporności na zakłócenia. Przerzutnik F10 synchronizuje sygnał przebiegiem /M1, co powoduje uniezależnienie się od wady mikroprocesora opisanej w p. 2. Bramki B28—B31 tworzą układ zapewniający generację impulsu RESET przy włączeniu komputera do sieci.

UWAGA! W celu zapewnienia prawidłowej pracy układu należy usunąć z komputera kondensator C27 (1 μF). Element ten, zapewniający start komputera bez dysku pamięciowego po włączeniu do sieci, uniemożliwiłby generację impulsu o zadanym czasie.

Uruchomienie układu dysku pamięciowego

Tak jak już wspomnieliśmy, ze względu na stopień złożoności układu nie zaleca się początkującym elektronikom prób uruchamiania opisanego urządzenia. Z tego też powodu zostaną podane jedynie ogólne wskazówki mające ułatwić proces instalacji dysku w komputerze, które mogą być łatwo zrozumiane przez doświadczonego elektronika-amatora.

Przed przystąpieniem do pracy zalecamy rozważenie sposobu konstrukcji mechanicznej urządzenia (wmontowane wewnątrz komputera lub w oddzielnej obudowie) w zależności od posiadanej wersji mikrokomputera (ZX Spectrum lub ZX Spectrum+). Szczególną uwagę radzimy zwrócić na problem zasilania, gdyż zasilacz wewnętrzny komputera ma bardzo mały zapas mocy. Następnie należy zaprojektować

i wykonać obwody drukowane dysku pamięciowego uwzględniając znane reguły projektowania obwodów techniki cyfrowej (należy pamiętać o ceramicznych kondensatorach blokujących zasilanie nie zaznaczonych na schemacie!). Następnie wprowadzić trzeba zmiany układowe posługując się schematem z rys. 2 (nie rozcinać zwor). Na wyprowadzeniu A8 układów pamięci podać logiczne 0 lub 1. Tak przerobiony komputer po włączeniu powinien zachowywać się normalnie: sprawdzić to można programami testującymi (np. TEST_PROG lub MEMTEST).

Dociekliwych Czytelników zdziwi zapewne fakt, że program testujący nie wykryje przekłamań, które powinny wystąpić przy niepełnym (siedmiobitowym) odświeżaniu. Otóż przekłamania te będą występować w części „niewidocznej" dla systemu (pamiętajmy, że pracując normalnie ma on dostęp zaledwie do 1/8 dostępnej pamięci).

Następnie należy dołączyć część sterującą pamięcią (schemat z rys. 3). Po obniżeniu RAMTOP-u poniżej wartości 32768 (ze względu na stos maszynowy) posługując się instrukcją OUT z poziomu BASIC-a należy sprawdzić, czy poszczególne banki zmieniają się (można to zrobić wpisując charakterystyczne wartości do określonych komórek) i przetestować każdy z nich programem testującym (tu uwaga: program TEST_PROG ze względu na swoją konstrukcję będzie wymagał wyzerowania systemu w celu powrotu: po wciśnięciu RESET należy ustawić RAMTOP poniżej wartości 32768, ustawić kolejny bank . instrukcją OUT 255, wczytać i uruchomić program testujący.

Kolejno dołączamy układ RESET (schemat na rys. 5). Jego test polega na oczywistym sprawdzeniu działania po włączeniu do sieci i skontrolowaniu prawidłowości synchronizacji. W tym celu naciskamy kilkunastokrotnie raz za razem przycisk RESET; po każdym naciśnięciu na ekranie powinien pojawić się czarny prostokąt, a następnie pokazać raport: © 1982 Sinclair Research Ltd. Jeżeli podczas wciskania przycisku na monitorze pojawią się kolorowe kwadraciki (nie zawsze — dlatego wciskamy wiele razy), będzie to świadczyć o wadliwej pracy układu synchronizującego.

Pamięć dodatkową (rys. 4) instalujemy i uruchamiamy w ostatniej kolejności. Jej działanie można bardzo łatwo sprawdzić z poziomu języka BASIC pisząc proste programy testujące. Jednakże należy pamiętać, że cykl pobrania rozkazu mikroprocesora Z80 jest krótszy od cyklu pobrania danej, dlatego też zaleca się wypełnienie pamięci zawartością #C9 (201), czyli instrukcją RET, a następnie wykonanie skoku (instrukcją RAND USR ...) do każdej z lokacji pamięci. Jeżeli nastąpi powrót z każdej lokacji (system nie „zawiesi się") można uznać pamięć za dobrą. Test taki jest bardzo prymitywny, zaleca się więc wykonanie innych, bardziej wyrafinowanych i tym samym wiarygodniejszych.

Dokładniejsze testy tak uruchomionego dysku pamięciowego można wykonać po wprowadzeniu programu obsługi, najprościej dokonując przesłań zawartości ekranu. Listing, opis obsługi programu, a także opis poszczególnych procedur systemowych zamieszczony zostanie w drugiej części artykułu.

Ciąg dalszy ze str. 8

przygotowywaniu dyskietek do pracy. W dalszej kolejności omówiono zlecenia stosowane do sprawdzania dysków i plików. Resztę książki polecamy bardziej zaawansowanym użytkownikom komputerów pracujących pod kontrolą systemu MS-DOS. Opisane są zlecenia do zmiany parametrów systemu i rozbudowy katalogu. Dokładnie przedstawiono wykorzystanie strumieni i pisanie makrodefinicji (pliki z rozszerzeniem BAT). Na zakończenie części będącej wiernym tłumaczeniem oryginału omówiono środki ułatwiające programowanie (debugery, linkery, edytory) oraz współpracę systemu MS-DOS z drukarką.



Oryginał książki ukazał się w 1984 r., a więc tłumaczenie radzieckie, aby być aktualne, musiało zawierać znaczne uzupełnienia treści. I tak opisano szczegółowo wersję DOS 3.2 i współpracę z dyskiem twardym, a także samą strukturę danych zapisanych na dyskietkach i dyskach. Zasygnalizowane są również wersje DOS 4.0 (system wielozadaniowy) i 5.0 (wersja na procesor 32-bitowy) oraz problemy z instalacją alfabetu rosyjskiego na komputerach IBM PC.

Podręcznik zawiera dużo przykładów z wyczerpującymi komentarzami i dwa znakomite dodatki: skrócony, alfabetyczny opis wszystkich zleceń z opisami komunikatów oraz opisy błędów generowane przez zlecenia MS-DOS.

Małgorzata Kalinowska-Iszkowska, Wacław Iszkowski: **Klucze do BASIC-u**, WNT 1987. Ostatnio polski rynek wydawniczy zalany został serią podręczników języka BASIC. W większości z nich autorzy wybierali sobie implementację tego języka na mniej lub bardziej popularny mikrokomputer. Książka „Klucze do BASIC-u" stanowi pewną nowość pod tym względem. Autorzy omówili w niej implementację BASIC-a na cztery popularne mikrokomputery: Amstrad, Apple II, IBM PC i ZX Spectrum. Wszystkie zostały potraktowane równorzędnie.



Już na początku autorzy zastrzegają się, że nie jest to podręcznik do nauki programowania lecz coś w rodzaju ściągawki. Zgrabne zestawienie instrukcji języka z umiejętnymi wyróżnieniami powodują, że książka w pełni zasługuje na to miano. Opis każdej implementacji rozpoczyna się od wyjaśnienia podstawowych elementów poleceń charakterystycznej dla danej wersji języka. Dalej zamieszczony jest tematycznie uporządkowany wykaz funkcji i poleceń, alfabetyczny wykaz komunikatów błędów.

Dzięki przemyślanemu układowi książka dobrze może służyć tym, którzy już coś wiedzą o programowaniu w BASIC-u, a jednocześnie potrzebują pomocy przy sprawdzeniu poprawnej postaci polecenia. Mankamentem tej książki jest brak opisu wersji języka dla Amstrada CPC-664 i CPC-6128 oraz Commodore 64 i 128. Mimo to nakłaniamy Czytelników do zakupu tej pozycji.

(kzjn)



ROLAND WACŁAWEK

TURBO-PASCAL I PRZERWANIA PROGRAMOWE w IBM PC/XT

Potrzeba odwołania się do języka maszynowego występuje w językach wysokiego poziomu najczęściej w dwóch przypadkach. Pierwszy z nich to potrzeba szczególnie szybkiej realizacji pewnych fragmentów algorytmu, np. związanych z operacjami ekranowymi. Bez napisania kawałka programu maszynowego się tu nie obejdzie, być może będą potrzebne nawet „sztuczki" programistyczne. Drugi przypadek to potrzeba skorzystania z usług systemu operacyjnego PC-DOS (MS-DOS) lub procedur zawartych w pamięci stałej BIOS.

Ponieważ przekazywanie parametrów odbywa się tu prawie wyłącznie za pośrednictwem rejestrów, to zapewnienie łączności procedury PC-DOS lub BIOS z programem w języku wysokiego poziomu wymaga z reguły napisania przynajmniej kawałka programu maszynowego, pośredniczącego w transferze parametrów. W znacznie lepszej sytuacji są tutaj użytkownicy TURBO-Pascala, który zawiera specjalne mechanizmy realizujące odwołania do PC-DOS lub BIOS bez potrzeby napisania chociażby jednego bajtu kodu maszynowego. Nie znaczy to jednak, że można obyć się bez podstawowej wiedzy o mikroprocesorze, a zwłaszcza o jego mechanizmie przerwań.

Wywoływanie procedur usługowych PC-DOS i BIOS odbywa się za pośrednictwem tzw. przerwań programowych. Przerwanie programowe polega na tym, że po wykonaniu specjalnego rozkazu maszynowego **INT** procesor wywołuje procedurę obsługi jednego z 256 możliwych przerwań procesora Intel 8088/6, zupełnie tak samo, jak gdyby przerwanie to zostało spowodowane przez przyczynę zewnętrzną. Jedynym argumentem rozkazu **INT** jest numer przerwania, którego wystąpienie należy w ten sposób „zasymulować".

W zasadzie rozkaz **INT** można by uznać za specyficzną formę wywołania odległej procedury. W stosunku do wywołania procedury rozkazem **CALL FAR** występują jednak dwie istotne różnice: **INT** zapisuje na stosie oprócz przesunięcia i segmentu powrotu także słowo stanu procesora, oprócz tego adres procedury obsługi jest pobierany z zarezerwowanego dla

każdego przerwania obszaru pamięci operacyjnej. W rozkazie **INT** wystarczy zatem podać numer przerwania. Wektory przerwań zajmują początkowe 1024 bajty pamięci operacyjnej: po 4 bajty na wektor (2 bajty przesunięcia + 2 bajty adresu segmentowego procedury obsługi). Ponieważ wektory przerwań są rozmieszczone w kolejności numeracji, to przesunięcie adresowe wektora przerwania nr **n** (n = 0...255) można wyznaczyć ze wzoru: **przesunięcie = 4*n.**

Procedury BIOS obsługują poszczególne, fizyczne urządzenia zewnętrzne, świadcząc usługi najbardziej podstawowe. Procedury te zostały podzielone na grupy, z których każda jest związana z odrębnym przerwaniem. Każda z tych grup obsługuje inne urządzenie zewnętrzne lub grupę urządzeń tej samej klasy. I tak przerwanie nr 16 (10H) obsługuje monitor, nr 19 (13H) — stacje pamięci dyskowej, nr 22 (16H) — klawiaturę, nr 23 (17H) — drukarkę, itd.

Procedury PC-DOS świadczą usługi o bardziej syntetycznym charakterze i — z niewielkimi wyjątkami — odwołują się raczej do urządzeń logicznych niż fizycznych. Prawie wszystkie funkcje usługowe PC-DOS są wywoływane za pośrednictwem jednego, jedynego przerwania nr 33 (21H). Żądana funkcja usługowa jest identyfikowana za pomocą numeru, który należy umieścić w rejestrze **AH** w chwili wywołania programu.

TURBO-Pascal pozwala uruchomić w sposób programowy dowolne spośród 256 przerwań. Służy do tego specjalna procedura standardowa **Intr**. Ma ona dwa parametry. Pierwszym z nich jest numer przerwania (0..255); musi być on stałą (nie wyrażeniem!). Drugim parametrem jest specjalny rekord o strukturze wiernie odwzorowującej wewnętrzny zestaw rejestrów mikroprocesora Intel 8088/6. Oto przykładowa definicja tego rekordu:

```
VAR rejestry: RECORD
                AX, BX, CX, DX, BP, SI, DI, DS, ES, flagi: integer
              END;
```

Przed wywołaniem funkcji **Intr** do odpowiednich pól tego rekordu należy wpisać takie wartości, jakie powinny znaleźć się w chwili wywołania w odpowiednich rejestrach. Po powrocie zawartość rejestrów można odczytać z odpowiadających im pól rekordu. Dotyczy to także rejestru flagowego, w którym można przetestować stan poszczególnych bitów-flag, używając pascalowego operatora **AND**. Żeby uniknąć nieporozumień: instrukcja, np.

```
rejestry.AX:= 13
```

nie powoduje bynajmniej wpisania stałej 13 do rejestru procesora **AX**, lecz wpisanie jej do określonej komórki pamięci operacyjnej. Zawartość poszczególnych komórek pamięci, odwzorowujących poszczególne rejestry, zostanie skopiowana do rejestrów mikroprocesora dopiero po wywołaniu procedury **Intr**, bezpośrednio przed spowodowaniem przerwania programowego. Natychmiast po powrocie z procedury obsługi tego przerwania zawartości rejestrów zostaną skopiowane do odpowiednich pól rekordu. Rekord nie reprezentuje więc bynajmniej bieżącego stanu rejestrów procesora, lecz tylko „migawkę" z chwili powrotu z wywołania.

Można użyć innej, bardziej odpowiedniej z punktu widzenia użytych argumentów definicji, pod warunkiem, że będzie ściśle zachowana kolejność odpowiednich pól-pseudorejestrów. Istotna jest przy tym tylko kolejność umieszczenia poszczególnych pseudorejestrów na liście, nie zaś ich nazwy:

```
VAR rejestry: RECORD
                AL, AH, BL, BH, CL, CH, DL, DH: byte;
                BP, SI, DI, DS, ES,         flagi: integer
              END;
```

Przy definiowaniu rejestrów jednobajtowych należy pamiętać, że rejestr przechowujący mniej znaczący (młodszy) bajt liczby musi zawsze poprzedzać w definicji bajt starszy.

Za najprostszy przykład zastosowania procedury **Intr** posłuży nam programowe sporządzanie kopii ekranu na drukarce. Aby wydrukować kopię bieżącej zawartości ekranu, wystarczy wywołać przerwanie nr 5 (przerwanie to jest uruchamiane m.in. po naciśnięciu kombinacji klawiszy **[Shift]** + **[PrtSc]**). Przy wywołaniu nie są potrzebne żadne parametry, a więc rekord pseudorejestrów spełnia funkcję czysto formalną. Obecność tego rejestru jest jednak niezbędna:

```
VAR rejestry: RECORD
                AX, BX, CX, DX, BP, SI, DI, DS, ES, flagi: integer
              END;
    ............
       Intr($5, rejestry);
```

Oto przykład bardziej złożony, w którym BIOS dostarczy nam kompletnej informacji o stanie klawiszy funkcyjnych klawiatury. Posłużymy się w tym celu funkcją nr 2 przerwania programowego BIOS nr $16. Numer funkcji należy wpisać do rejestru **AH**. Po powrocie z procedury obsługi poszczególne bity rejestru **AL** informują o stanie klawiszy funkcyjnych: **[Shift]**, **[Ctrl]**, **[Alt]**, **[CapsLock]**, **[NumLock]**, **[ScrollLock]** i **[Insert]**:

```
VAR wyniki: RECORD
              AL, AH                             : Byte;
              BX, CX, DX, BP, SI, DI, DS, ES, flagi: Integer
            END;
    AL_pop: Byte;
BEGIN ClrScr;
  Write('Naciskaj klawisze funkcyjne');
  REPEAT AL_pop:= wyniki.AL;
    wyniki.AH:= 2;
    Intr($16, wyniki);
    IF wyniki.AL<>AL_pop THEN
      WITH wyniki DO
        BEGIN
          GotoXY(1, 25); DelLine;
          GotoXY(1, 25); IF (AL AND $80)>0 THEN Write('Insert');
          GotoXY(13,25); IF (AL AND $40)>0 THEN Write('CapsLock');
          GotoXY(25,25); IF (AL AND $20)>0 THEN Write('NumLock');
          GotoXY(37,25); IF (AL AND $10)>0 THEN Write('ScroLock');
          GotoXY(49,25); IF (AL AND $08)>0 [illegible] Write('Alt');
          GotoXY(61,25); IF (AL AND $04)>0 [illegible] Write('Ctrl');
          GotoXY(73,25); IF (AL AND $03)>0 THEN Write('Shift')
        END
  UNTIL (wyniki.AL [illegible] $03)=$03
END.
```

Testowanie poszczególnych bitów odbywa się przez wykonanie iloczynu logicznego odczytanego bajtu stanu klawiatury z innym bajtem-maską, w którym jest ustawiony tylko wybrany bit. Program na bieżąco wskazuje u dołu ekranu stan wyszczególnionych klawiszy i kończy pracę w chwili równoczesnego wciśnięcia lewego i prawego klawisza **[Shift]** (fakt ten jest rozpoznawany po równoczesnym ustawieniu dwóch najmłodszych bitów; patrz artykuł w „InforMiku" 2/87).

W zestawie standardowych narzędzi graficznych TURBO-Pascala dokuczliwie odczuwany jest brak funkcji, dostarczającej informacji o stanie poszczególnych punktów ekranu w trybie graficznym, a więc pozwalającej programowi „widzieć" ekran. Ponieważ jednak operację testowania ekranu realizuje jedna z funkcji BIOS, zdefiniowanie odpowiedniej funkcji we własnym zakresie nie będzie trudne.

Testowanie ekranu realizuje funkcja nr 13 przerwania nr 16 (10H), obsługującego operacje ekranowe. Funkcja wymaga przekazania dwóch parametrów — współrzędnych testowanego punktu ekranu: **X** w rejestrze **CX** i **Y** w rejestrze **DX**. Po powrocie z wywołania rejestr AL zawiera kod barwy danego punktu ekranu. W przypadku karty CGA w trybie wysokiej rozdzielczości 640*200 punktów możliwe są wartości 0 i 1, w trybie wielobarwnym 320*200 punktów — 4 wartości 0 do 3. Oto gotowa definicja funkcji **Punkt**, dostarczającej kodu barwy we wskazanym punkcie ekranu:

```
FUNCTION Punkt(x, y:Integer): Byte;
VAR wyniki: RECORD
              AL, AH                        : Byte;
              BX, CX, DX, BP, SI, DI, DS, ES, flagi: Integer
            END;
BEGIN
    wyniki.AH:= 13;
    Wyniki.CX:= x; Wyniki.DX:= y;
    Intr($10, wyniki);
    Punkt:= wyniki.AL
END;
```

Ponieważ wywołania systemu PC-DOS za pośrednictwem przerwania 21H zdarzają się dość często, TURBO--Pascal oferuje tu specjalną procedurę standardową **MsDos**. Działa ona identycznie jak **Intr**, ale wymaga tylko pojedynczego parametru w postaci omówionego rekordu (numer funkcji należy wcześniej przypisać pseudorejestrowi **AH**).

Oto prosty, lecz użyteczny przykład zastosowania procedury **MsDos**, pozwalający ustalić rozmiar wolnego obszaru na wybranym dysku. Informacji takiej udzieli funkcja usługowa PC-DOS nr 54 (36H). Funkcja wymaga podania tylko jednego parametru, a mianowicie numeru stacji. Wartość 0 oznacza stację domniemaną (dyżurną), wartość 1 — stację ⟨A⟩, 2 — stację ⟨B⟩, itd. Po powrocie w rejestrach znajduje się szereg użytecznych informacji. **BX** zawiera liczbę wolnych jednostek allokacji (ang. *cluster*), **DX** — ogólną pojemność dysku, wyrażoną w jednostkach allokacji, **CX** — liczbę bajtów w pojedynczym sektorze, natomiast **AX** — liczbę sektorów, przypadających na jednostkę allokacji (w przypadku podania numeru nie istniejącej stacji dysków **AX** zawiera liczbę $FFFF). Chcąc ustalić rozmiar pamięci w bajtach, wystarczy pomnożyć liczbę jednostek allokacji przez liczbę sektorów w jednostce i liczbę bajtów w sektorze:

```
VAR Symbol:   Char;
    Rejestry: RECORD
                AX, BX, CX, DX, BP, SI, DI, DS, ES, flagi: Integer
              END;
BEGIN Write('Podaj symbol stacji: '); Readln(Symbol);
      Symbol:= UpCase(Symbol);
      WITH Rejestry DO
        BEGIN
          AX:= $3600;
          DX:= Ord(Symbol)-Ord('@');
          MsDos(Rejestry);
          IF AX<>$FFFF
             THEN Writeln('Wolny obszar =',
                          1.0*AX*BX*CX:8:0, ' bajtow')
             ELSE Writeln('Ta stacja nie istnieje !')
        END
END.
```

Program pyta o symbol stacji, wczytuje z klawiatury pojedynczy znak, następnie sprawdza, czy znak nie jest małą literą, a jeśli tak — zamienia ją na dużą. Potem znak jest zamieniany na numer stacji. Jeżeli podano znak 'C' (ASCII 64), operacja będzie dotyczyć stacji dyżurnej (domniemanej), litera '**A**' (ASCII 65) — odpowiada stacji ⟨A⟩, itd. Wyjaśnienia wymaga stała 1.0 przed iloczynem zawartości pseudorejestrów. Stała ta forsuje (wymusza) obliczenia na liczbach typu **Real**. Otóż pseudorejestry te są zmiennymi typu całkowitego, w związku z czym TURBO-Pascal użyje operacji całkowitoliczbowych, ignorując ewentualny nadmiar arytmetyczny. Jeżeli jednak już na początku wyrażenia wystąpi liczba typu **Real**, to wszystkie kolejne obliczenia będą prowadzone także na liczbach typu **Real**.

Jeżeli w programie pascalowym potrzebny jest np. bieżący czas, to można uzyskać go tylko odwołując się do systemu PC-DOS. Uczyniono tak w poniższym programie, wyświetlającym na ekranie bieżący czas systemowy. Czasu dostarcza funkcja usługowa PC-DOS nr 44 ($2C). Przed wywołaniem tej funkcji należy do pseudorejestru **AH** wpisać kod funkcji. Po powrocie z wywołania rejestr **CH** zawiera godziny, **CL** — minuty, **DH** — sekundy, a **DL** — setne części sekundy. Procedura **MsDos** jest wywoływana w oddzielnej funkcji **Czas__systemowy**, przekształcającej otrzymane z systemu dane liczbowe na postać tekstową:

```
TYPE rejestry = RECORD
                  AL, AH, BL, BH, CL, CH, DL, DH:  byte;
                  BP, SI, DI, DS, ES, flagi    : integer
                END;
     tekst     = STRING[20];

     FUNCTION Czas_systemowy: tekst;
     VAR   wyniki          :  Rejestry;
           godz, min, sek: STRING[2];
     BEGIN wyniki.AH:= $2C;
           MsDos(wyniki);
           WITH wyniki DO BEGIN
              Str(CH, godz); Str(CL, min); Str(DH, sek);
           END;
           Czas_systemowy:= godz + ':' + min + ':' + sek
     END;

BEGIN  Writeln('Aktualny czas :  ', Czas_systemowy)
END.
```

Po odczytaniu z systemu operacyjnego czasu, przekazanego w poszczególnych pseudorejestrach, wystarczy zamienić liczbę godzin, minut i sekund na odpowiedni łańcuch i dodać stosowne separatory.

Na zakończenie przeanalizujmy bardziej złożone, lecz nader przydatne w praktyce zastosowanie procedury **MsDos**. Wczytamy mianowicie do programu pascalowego zawartość katalogu dyskowego.

Nowoczesne programy powinny dbać o komfort użytkownika. Jeśli np. program wymaga podania plików dyskowych, to najlepiej wyświetlić na ekranie katalog dyskowy i umożliwić wybór pliku np. przez podświetlenie jego nazwy. TURBO-Pascal nie daje jednak niestety możliwości bezpośredniego odczytu katalogu dyskowego w celu np. przypisania nazw poszczególnych plików zmiennym łańcuchowym. Szkoda, gdyż potrzeba automatycznej analizy katalogu dyskowego występuje w wielu zastosowaniach. Mimo to stworzymy program w TURBO-Pascalu, który odczyta do tablicy łańcuchowej wszystkie pliki, których nazwy są zgodne z podanym szablonem, uporządkuje je alfabetycznie i wyprowadzi na ekran.

Aby uzyskać w TURBO-Pascalu dostęp do katalogu dyskowego, trzeba posłużyć się bezpośrednio odpowiednimi funkcjami usługowymi PC-DOS. Najprościej użyć funkcji 4EH i 4FH, przedtem trzeba będzie jednak przekazać systemowi operacyjnemu adres obszaru DTA (ang. *Disk Transfer Area*). DTA to dowolnie wybrany obszar pamięci operacyjnej, pośredniczący w wymianie danych o większej objętości przy operacjach dyskowych. W omawianym przypadku w obszarze DTA system operacyjny będzie przekazywał dane, odczytane z kolejnych pozycji katalogu dyskowego.

Nasz obszar DTA będzie zwyczajną tablicą znakową, liczącą 43 bajty — dokładnie tyle, ile wymagają wspomniane funkcje. Do przekazania systemowi operacyjnemu adresu DTA użyjemy funkcji nr 26 (1AH). Adresu tego, tzn. adresu tablicy, dostarczą nam funkcje standardowe TURBO-Pascala **Seg** i **Ofs**. Program najpierw pyta o szablon (szkielet) nazwy. Można w nim użyć znaków zastępczych '*' i '?'. Szablon postaci *,* spowoduje wylistowanie całego katalogu. Nasz program akceptuje katalogi zawierające maksymalnie 512 plików — tyle łańcuchów może pomieścić tablica **Pliki**.

Właściwą analizę katalogu realizują dwie kolejne funkcje usługowe PC-DOS. Funkcja nr 78 (4EH) znajduje pierwszą pozycję katalogu dyskowego, zgodną z podanym szablonem. Natomiast funkcja nr 79 (4FH) odnajduje najbliższą, kolejną pozycję katalogu, zgodną z szablonem przekazanym w ostatnim wywołaniu funkcji nr 78. Tak więc analizę katalogu należy rozpocząć zawsze od wywołania funkcji 78, przekazując przy okazji systemowi PC-DOS szablon nazwy pliku. Jeżeli analizowany katalog nie jest katalogiem domniemanym (dyżurnym), to specyfikację tę można poprzedzić ścieżką do katalogu, zgodnie z ogólnymi zasadami w systemie PC-DOS. Trzeba jedynie pamiętać, że PC-DOS wymaga, aby specyfikacja ścieżki była zakończona bajtem 0 (ASCII 0). W TURBO-Pascalu format łańcuchów nie przewiduje standardowego znaku #0 na końcu łańcucha, należy więc znak ASCII 0 dołączyć do łańcucha wprost w instrukcji przypisania. Wywołując funkcję nr 79, nie trzeba już podawać szablonu, gdyż w dalszym ciągu obowiązuje szablon, podany w funkcji nr 78.

```
TYPE Nazwa_pliku: STRING[12];
VAR  Szablon     : STRING[64];
     Ost_plik, i : Integer;
     Pliki       : ARRAY[1..512] OF Nazwa_pliku;
     DTA         : ARRAY[0..42] OF Char;
     Rejestry    : RECORD
                     AL, AH                           : byte;
                     BX, CX, DX, BP, SI, DI, DS, ES, flagi: integer
                   END;

  PROCEDURE Dopisz_nazwe_pliku;
  VAR   nr_zn: Integer;
  BEGIN
    Ost_plik:= Ost_plik+1;   nr_zn:= 30;
    Pliki[Ost_plik]:= '';
    WHILE DTA[Nr_zn]<>#0 DO
          BEGIN
             Pliki[Ost_plik]:= Pliki[Ost_plik]+DTA[nr_zn];
             nr_zn:= nr_zn+1
          END;
  END;

BEGIN
     Write('Podaj szablon nazwy pliku: ');
     Readln(Szablon);
     Szablon:= Szablon+#0;
     Ost_plik:= 0;
     WITH Rejestry DO
       BEGIN
         DS:= Seg(DTA);
         DX:= Ofs(DTA);
         CX:= 0;
         AH:= $1A;
         MsDos(Rejestry);
```



```
         DS:= Seg(Szablon[1]);
         DX:= Ofs(Szablon[1]);
         AH:= $4E;
         MsDos(Rejestry);

         IF AL=0 THEN
           REPEAT
              Dopisz_nazwe_pliku;
              AH:= $4F;
              MsDos(Rejestry);
           UNTIL AL<>0;
         IF AL=2
             THEN Writeln('Brak plikow zgodnych z szablonem')
             ELSE FOR i:=1 TO Ost_plik DO Writeln(Pliki[i])
       END
END
```

Zauważ, że przy przekazywaniu funkcji 78 specyfikacji szablonu podano nie adres łańcucha, lecz jego pierwszego znaku. W TURBO-Pascalu pierwszy bajt obszaru pamięci, przechowywującego łańcuch, zawiera bowiem licznik znaków w tym łańcuchu. Właściwa treść łańcucha zaczyna się dopiero od drugiego bajtu. Podając adres łańcucha, wskazalibyśmy więc na pierwszy bajt, przechowujący bieżącą długość łańcucha.

Jeżeli kolejny łańcuch zgodny z podanym wcześniej szablonem został odnaleziony i przekazany do obszaru DTA, to po powrocie rejestr **AL** zawiera 0. Jeżeli zawartość AL jest różna od 0, znaczy to, iż wystąpił błąd. W naszym przypadku jedyną przyczyną błędu może być nieodnalezienie plików zgodnych z szablonem. Jeżeli **AL = 2**, znaczy to, że nie stwierdzono ani jednego pliku zgodnego z szablonem (błąd ten może wystąpić wyłącznie w funkcji nr 78), natomiast **AL = 18** oznacza, że nie ma już więcej plików zgodnych z szablonem (poprzednio wystąpił co najmniej 1 plik zgodny z szablonem).

PC-DOS przekazuje do DTA specyfikację pliku w formacie niezgodnym z wewnętrznym formatem zapisu łańcuchów w TURBO-Pascalu, po prostu jako ciąg znaków zakończony znakiem o kodzie 0. Zadaniem procedury **DopiszPlik** jest m.in. analiza tego ciągu i przekształcenie go na łańcuch w formacie TURBO-Pascala. Najprościej osiągnąć to, dołączając kolejne znaki do początkowo pustego łańcucha, aż do napotkania znaku o kodzie 0.

Możliwość łatwego dostępu do wszystkich funkcji BIOS i DOS w połączeniu z dużą efektywnością generowanego kodu czynią z TURBO-Pascala narzędzie, przydatne także do tworzenia oprogramowania systemowego.

MAGNETOFON KOMPUTERA COMMODORE C64



WOJCIECH ŻUREK

Wśród licznej rzeszy użytkowników komputera C64 większość jako pamięć zewnętrzną stosuje magnetofon. Konstrukcja firmowego magnetofonu oraz przyjęty przez producenta sposób zapisu sygnału cyfrowego dają dużą pewność współpracy komputera z pamięcią kasetową. Co więcej, powszechnie stosowany program TURBO TAPE przyspieszający około dziesięciokrotnie zapis i odczyt na taśmie magnetofonowej, nie wpływa na jakość tej współpracy. Ta pozytywna opinia nie oznacza bynajmniej, że ten sprzęt szczególnie po pewnym okresie eksploatacji lub w specyficznych warunkach nie zacznie zawodzić. W artykule tym postaram się przekazać wiedzę potrzebną do ewentualnej regulacji czy naprawy uszkodzonego magnetofonu oraz przedstawię pewne proste układy, które pozwoliły mi na zwiększenie komfortu pracy z tym typem pamięci zewnętrznej.

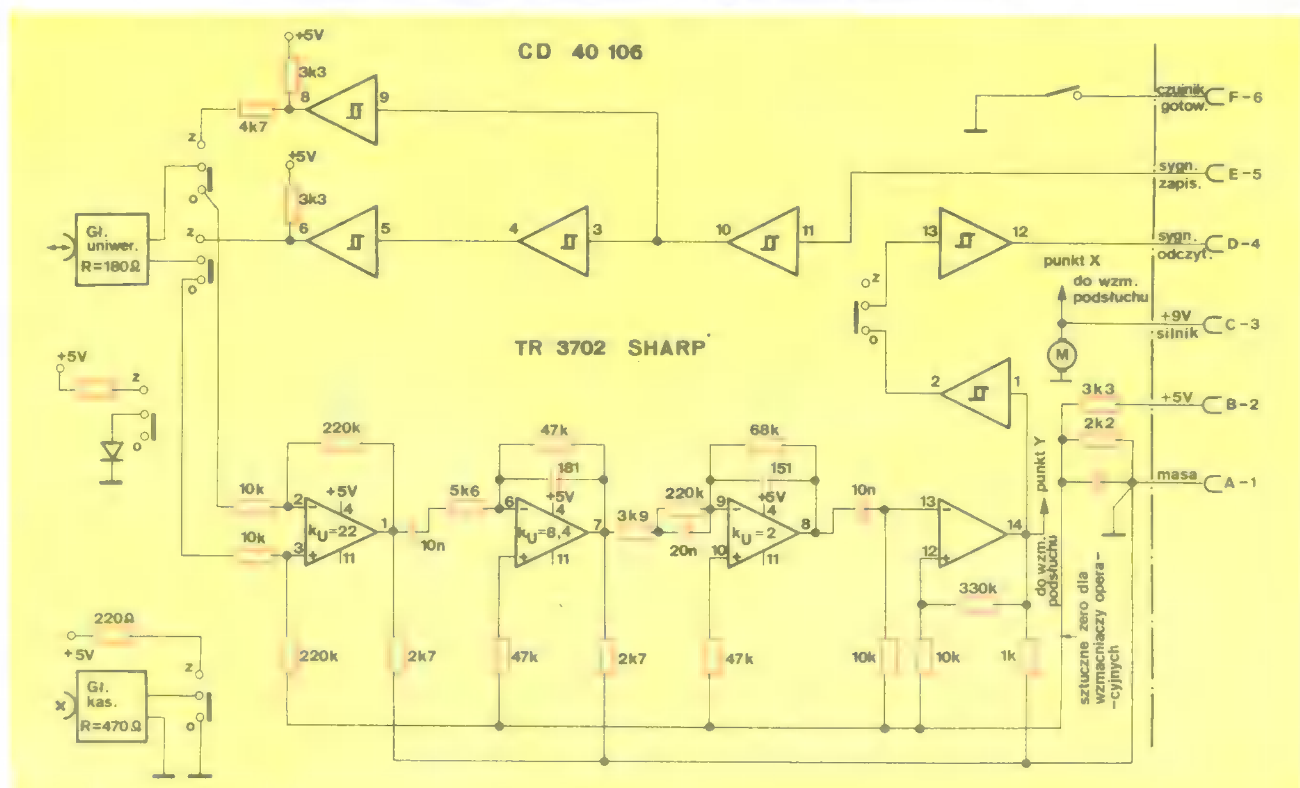
Wyprowadzenie	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Wartość napięcia stałego	2,05	2 V	2 V	5V	2V	2V	2V	2V	2V	2V	0V	2V	2V	1,7V
Kształt przebiegu	∿	—	—	+ zasil.	—	—	∿	⊓⊔	—	—	masa zasil.	—	—	⊔⊓
Amplituda	230 mV	—	—		—	—	2 V	4V	—	—		—	—	4 V

Tab. 1. Wartości napięć stałych oraz charakterystyczne przebiegi na wyprowadzeniach układu scalonego TR 3702

Trochę teorii

Sygnał zapisywany na taśmie ma postać fali prostokątnej, o zmiennym czasie trwania impulsów (*Pulse Position Modulation*). Wśród impulsów możemy wyróżnić impulsy K (o krótkim czasie trwania), S (o czasie średnim) oraz D (o czasie długim). Informacja kodowana jest za pomocą czterech następujących po sobie impulsów, przy czym jej znaczenie zależy od ich długości i kolejności.

Rys. 1. Schemat ideowy magnetofonu PM-16 do komputera Commodore C64



Na przykład czwórka DDSS oznacza początek bajtu, w obrębie bajtu czwórka SSKK oznacza jedynkę, a kombinacja KKSS zero. Zgodnie z tym zapis na taśmie bajtu odpowiadającego literze „A" przyjmie postać (kod litery A = $41):

DDSS	SSKK	KKSS	KKSS	KKSS
Identyfikator	1	0	0	0
Bit	0	1	2	3

KKSS	KKSS	SSKK	KKSS	SSKK
0	0	1	0	1
4	5	6	7	sumy kontrolnej

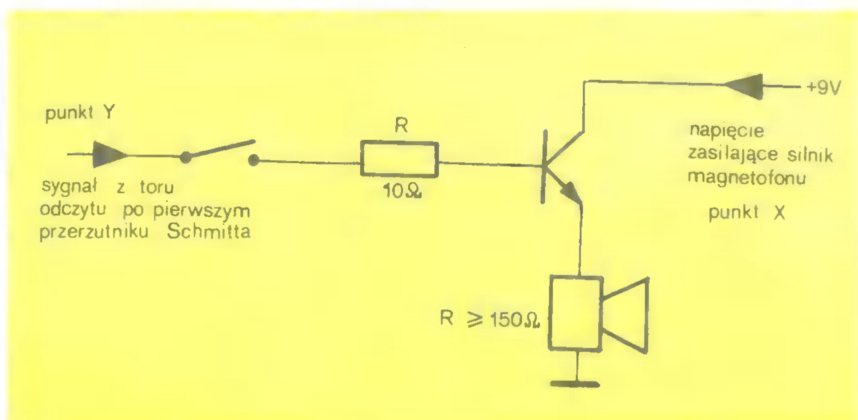
Średnia częstotliwość zapisywanego sygnału prostokątnego wynosi około 2,5 kHz dla zapisu standardowego oraz około 5 kHz przy zapisie za pomocą programu TURBO TAPE.

Likwidujemy usterki

Do podstawowych, najczęściej spotykanych usterek powodujących błędny odczyt z taśmy magnetofonowej (pomijamy oczywiście błędy spowodowane uszkodzeniem taśmy oraz błędy, które są następstwem zabezpieczania programów przed kopiowaniem) należy zabrudzenie głowicy oraz jej rozregulowanie (zmiana skosu głowicy). W pierwszym przypadku należy głowicę przemyć tamponem nawilżonym spirytusem (denaturatem, izopropanolem) zwracając przy tym uwagę na to, aby nie zarysować czoła głowicy. W drugim trzeba wykonać regulację skosu głowicy. Nie musimy przy tym używać rozbudowanego zestawu pomiarowego. Wystarczy wykorzystać napisany w tym celu program o nazwie TAPE HEAD JUSTAGE. Po załadowaniu i uruchomieniu programu (jeżeli to będzie konieczne to z pożyczonego magnetofonu) uruchamiamy uszkodzony magnetofon i za pomocą śruby regulacyjnej przy głowicy ustawiamy jej skos tak, aby punkty wyświetlane na monitorze były jak najbardziej skupione i dawały obraz paska. Powinien on całkowicie zawierać się pomiędzy zależnymi od standardu liniami pomocniczymi. Przy pewnej wprawie i zastosowaniu układu lub programu dającego możliwość podsłuchu (niektóre wersje programu TURBO) możemy też głowicę ustawić „na słuch". Regulujemy głowicą tak, by otrzymać czysty, zawierający jak najwięcej wysokich tonów dźwięk. Sposób ten jednak nie jest tak dokładny jak powyżej opisany.

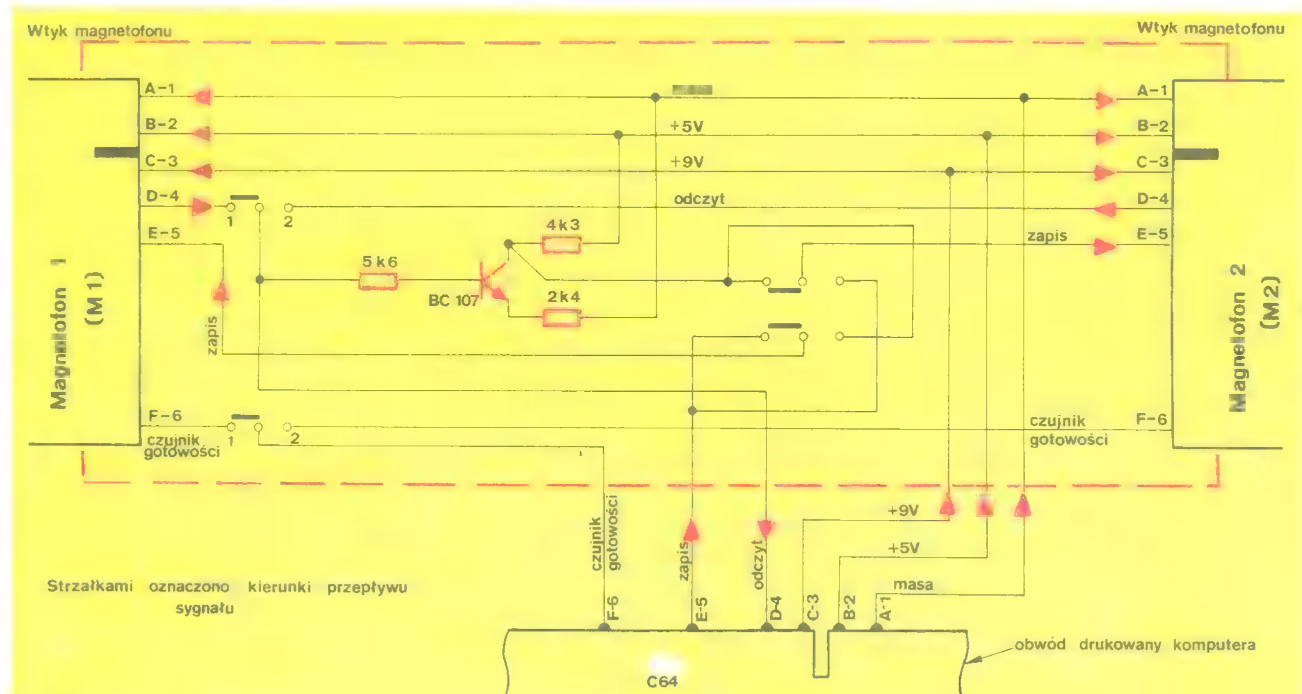
Jeżeli wymienione czynności nie dadzą rezultatu znaczy to, że uszkodzenie jest innego typu. Zmuszeni więc jesteśmy ingerować w układ wewnętrzny magnetofonu (niestety czasem też komputera). Zapoznajmy się więc z jego schematem wewnętrznym na przykładzie magnetofonu typu PM-C16. Jest to tajwańska kopia oryginalnego magnetofonu firmy Commodore (model 1530), popularna dla tego rodzaju sprzętu.

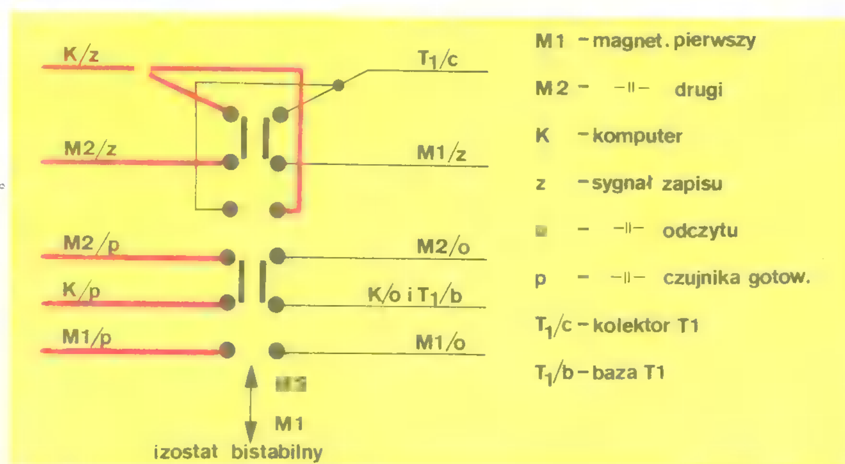
Analizując schemat pokazany na rys. 1, zauważamy typowe i bardzo uproszczone podukłady magnetofonowe. Tor zapisu składa się z czterech inwerterów CMOS, przy czym w celu poprawy zbocza impulsu zastosowano inwertery Schmitta. Pierwszy z nich jest stopniem separującym, natomiast trzy pozostałe połączono w ten sposób, by zapewnić zmianę kierunku prądu głowicy uniwersalnej przy zmianie poziomu sygnału wejściowego. Wraz ze zmianą kierunku prądu w głowicy zmienia się kierunek wytwarzanego przez nią pola, a co za tym idzie kierunek uporządkowania dipoli taśmy magnetycznej. Wartość natężenia prądu głowicy w stanie ustalonym jest ograniczona za



Rys. 2. Jednostopniowy wzmacniacz podsłuchu

Rys. 3. Układ podłączenia dwóch magnetofonów do komputera Commodore C64





Rys. 4. Przełącznik wyboru magnetofonu źródłowego

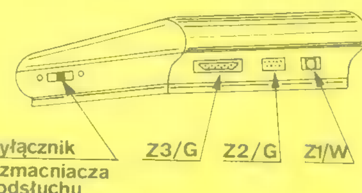
pomocą opornika. W magnetofonie zrezygnowano z generatora prądu podkładu. Jest on zbyteczny, gdyż szumy przy zapisie i odczycie sygnału cyfrowego nie odgrywają istotnej roli, a kasowanie starego zapisu może być dokonywane polem stałym. W związku z tym na czas zapisu głowica kasująca podłączona jest do napięcia stałego +5 V za pomocą opornika ustalającego prąd kasowania, natomiast na czas odczytu zwierana do masy. Sprawdzenie i wychwycenie ewentualnych uszkodzeń toru zapisu jest proste. Polega ono przede wszystkim na wizualnym sprawdzeniu szczeliny i czoła głowicy, a potem na pomiarze wartości prądu kasowania (około 70 mA) oraz prądu głowicy uniwersalnej. Prąd ten powinien mieć wartość około 100 mA i zmieniać kierunek przy zmianie sygnału wejściowego. Sprawdzenia możemy dokonać w sposób statyczny zwierając wejście zapisu do masy, a następnie do +5 V.

UWAGA! Wejście to musi być koniecznie odłączone od komputera!!! Uszkodzone elementy należy zastąpić sprawnymi, przy czym w miejsce nie produkowanego u nas układu scalonego CD 40106 możemy zastosować układ MCY 74069 (zwykły inwerter CMOS), a w miejsce głowicy uniwersalnej głowicę produkcji krajowej o takiej samej konstrukcji mechanicznej i podobnej oporności korygując doświadczalnie jej prąd. W konstrukcji próbnej zupełnie poprawnie funkcjonowała głowica od polskiego magnetofonu MK-125 oraz wymieniony układ scalony.

Najbardziej rozbudowanym układem magnetofonu jest tor odczytu. Ze względu na niskie napięcie zasilania konieczne jest tu stosowanie wzmacniaczy niskonapięciowych. W konstrukcji fabrycznej zastosowano poczwórny wzmacniacz operacyjny TR 3702. Pierwszy stopień, to typowy wzmacniacz różnicowy. Z jego wyjścia sygnał podawany jest na dwustopniowy wzmacniacz korekcyjny, a potem kolejno na trzy przerzutniki Schmitta. Wzmocnienie pierwszego stopnia wynosi około 22 razy, drugiego około 8,5 raza, a trzeciego — około 2 razy. Dokładne sprawdzenie tego toru jest trudne, wymaga odpowiedniej aparatury. Jednak pewne wnioski możemy wyciągnąć, mierząc napięcia stałe oraz obserwując przebiegi w poszczególnych punktach układu. W celu ułatwienia tej czynności w tabeli 1 przedstawiono wielkości zmierzone w sprawnym torze odczytu. W przypadku konieczności wymiany wzmacniacza operacyjnego, w miejscu układu TR 3702 można zastosować cztery wzmacniacze niskonapięciowe typu SFC 2861 lub inne o podobnych parametrach (zmiana typu układu scalonego pociąga za sobą zmianę obwodu drukowanego).

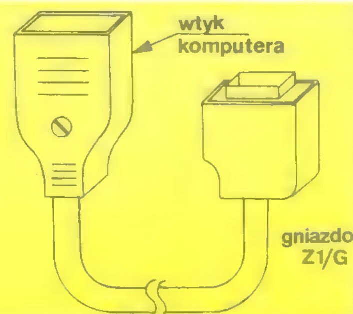
Pozostałe układy magnetofonu (wskaźnik nagrywania — dioda LED, czujnik naciśnięcia klawiszy, dzielnik napięcia zasilania) są bardzo proste, a sprawdzenie

Styk nr	Adres	Przesyłany sygnał	
1	Z3/G - 20	A-1 / K	masa
2	Z3/G - 21	B-2 / K	+5V
3	Z3/G - 22	C-3 / K	+9V
4	Z3/G - 23	D-4 / K	odczyt
5	Z3/G - 24	E-5 / K	zapis
6	Z3/G - 25	F-6 / K	czujnik
7	NC	sygnały komputera	
8	NC	przez to złącze	
9	NC	podawane są na Z3	



Tab. 2. Połączenia wtyku Z1/W mocowanego w magnetofonie, do którego podłącza się przewód z komputera

Styk nr	Adres	Przesyłany sygnał	
1	WK–A–1	A-1/K	masa
2	WK–B–2	B-2/K	+5V
3	WK–C–3	C-3/K	+9V
4	WK–D–4	D-4/K	odczyt
[illegible]	WK–E–5	E-5/K	zapis
[illegible]	WK–F–6	F-6/K	czujnik
7	NC	sygnały komputera	
[illegible]	NC	przez to złącze	
9	NC	podawane są [illegible] Z3	



Tab. 3. Połączenia gniazda Z1/G zakończającego sznur łączący z komputerem od strony magnetofonu

ich nie powinno nastręczać większego kłopotu.

Osobny problem stanowi prędkość przesuwu taśmy. Zbyt duże nierównomierności przesuwu lub zbyt odbiegająca od znamionowej prędkość mogą, na skutek przyjętego sposobu kodowania, prowadzić do błędów odczytu. Uszkodzenia należy szukać w mechanice magnetofonu, w układzie stabilizacji prędkości lub w silniku. Należy przy tym pamiętać, iż silnik zasilany jest z komputera oddzielnym napięciem (+9 V) kontrolowanym przez pierwszą komórkę pamięci. Zatrzymać go możemy wykonując zlecenie

```
POKE 191,1 : POKE 1, PEEK(1) OR 32
```

a uruchomić przez

```
POKE 191,1 : POKE 1, PEEK(1)
                        AND 32
```

Chciałbym zwrócić uwagę na jeszcze jeden element układu, który wywoływał w działaniu mojego magnetofonu najdziwniejsze objawy. Na skutek zabrudzenia styków przełącznika zapis-odczyt, magnetofon nie odtwarzał lub nie nagrywał. Dopiero po wyczyszczeniu i nasmarowaniu smarem silikonowym jego styków objawy te ustąpiły.

Podsłuch

Dużym udogodnieniem przy pracy z magnetofonem jest podsłuch. Programy dające tę możliwość (jak np. niektóre wersje TURBO, w których efekty dźwiękowe związane z wczytywaniem programu podawane są na wyjście audio i zobrazowane na monitorze) rozwiązują problem tylko częściowo. Celowe więc staje się wykonanie prostego układu elektronicznego. Swoje zadanie w zupełności spełnia układ przedstawiony na rys. 2. Jest to jednostopniowy wzmacniacz, który bez trudu można umieścić we wnętrzu magnetofonu na oddzielnej płytce lub przy wykorzystaniu nie obsadzonych fragmentów obwodu fabrycznego. Przy jego wykonaniu największe kłopoty może sprawić zdobycie odpowiedniego głośniczka. W rozwiązaniu modelowym wykonano go ze starej słuchawki aparatu dla słabo słyszących. W celu poprawy skuteczności przetwarzania odsłonięto membranę przez zeszlifowanie osłony plastikowej. Możliwe jest też użycie słuchawki telefonicznej

Styk nr	Adres	Przesyłany sygnał
1	Z3/G – 14	A-1/M2 masa
2	Z3/G – 15	B-2/M2 +5V
3	Z3/G – 16	C-3/M2 +9V
4	Z3/G – 17	D-4/M2 odczyt
5	Z3/G – 18	E-5/M2 zapis
6	Z3/G – 19	F-6/M2 czujnik
7	NC	sygnały magn. M2
8	NC	przez to złącze
9	NC	podawane są na Z3

Tab. 4. Połączenia gniazda Z2/G mocowanego w przerabianym magnetofonie, służące do podłączenia drugiego magnetofonu

o oporności 150 omów. W tym jednak wypadku mogą wystąpić kłopoty z umieszczeniem układu wewnątrz magnetofonu. Przy zastosowaniu innego typu głośniczka należy zwrócić uwagę na to, by jego oporność była nie mniejsza od 100 omów. Jest to konieczne ze względu na to, że całość układu zasilana jest z wewnętrznego napięcia komputera +9 V zasilającego silnik magnetofonu, a dodatkowe obciążenie tego źródła nie powinno przekraczać 100 mA (mając na uwadze obciążenie go ewentualnie jeszcze drugim silnikiem). Sygnał dostarczany do wzmacniacza pobierany jest z wyjścia pierwszego przerzutnika Schmitta poprzez opornik ograniczający pobór mocy. W zależności od potrzeb sygnał ten może być podawany również przez mikrowyłącznik.

Kopiujemy programy

Następny układ pokazany na rys. 3 ma na celu ułatwienie pracy przy kopiowaniu programów. Pozwala on na jednoczesną pracę komputera z dwoma magnetofonami (w sensie równoczesnego nagrywania i odtwarzania) bez konieczności zmiany kasety czy przełączania magnetofonów. Istnieje przy tym możliwość wyboru magnetofonu (za pomocą przełącznika typu isostat), z którego wczytywany jest program i zarazem kontrolowany sygnał o stanie przełączników. Za pomocą tego zestawu w prosty sposób możemy kopiować programy łącznie z ich pełnymi nazwami, co przy programach o objętości większej niż pojemność pamięci dostępnej dla BASIC-a nie jest możliwe. Jest to rozwinięcie konstrukcji przedstawionej w numerze 3—4/1986 czasopisma „Bajtek".

C64 połączony jest z magnetofonem za pomocą sześciu przewodów. Trzy z nich dostarczają napięć zasilających: napięcie +5 V do zasilania elektroniki, kontrolowane napięcie +9 V do zasilania silnika i masa. Trzy pozostałe przewody służą do transmisji sygnału (standard TTL) odczytywanego z magnetofonu, zapisywanego na nim oraz informującego o stanie klawiszy (masa — magnetofon włączony, przerwa — wyłączony). By móc pracować na dwu magnetofonach równocześnie musimy do obu dostarczyć napięć zasilających, z jednego z nich pobrać odczytywany sygnał, a do drugiego wysłać sygnał, który ma być zapisany. Musimy też dostarczyć do komputera informacji o gotowości magnetofonu do podjęcia pracy. Normalnie nie istnieje możliwość jednoczesnego pobierania i wysyłania programu przez komputer. Sygnał do zapisu otrzymujemy z odtwarzanego sygnału (a zarazem wczytywanego przez komputer) w oddzielnym układzie elektronicznym. Układ ten zmienia fazę sygnału co jest konieczne dla poprawnej pracy. Wykonano go korzystając z inwertera z tranzystorem. Do budowy przełącznika zastosowano przełącznik typu isostat dwusekcyjny (cztery styki przełączalne). Jego styki (rys. 4) realizują następujące funkcje: pierwszy zmienia źródło sygnału pobieranego przez komputer i jednocześnie przez inwerter, drugi — źródło informacji o gotowości do pracy, a dwa ostatnie rozdzielają sygnały zapisu. Istnieją dwa sygnały zapisu: jeden z komputera, który powinien trafiać zawsze do magnetofonu źródłowego, a drugi z inwertera, który ma być dostarczony do drugiego magnetofonu. **Linie, na których one występują nie mogą być ze sobą połączone, gdyż możemy uszkodzić komputer!!!** Nie wolno nam połączyć wyjścia mikroprocesora z wyjściem inwertera, gdyż w wypadku wymuszania przez nie przeciwnych stanów, możemy przeciążyć elementy!

Układ możemy zmontować na obwodzie drukowanym nanosząc na nim oprócz potrzebnych ścieżek rysunki wtyków do podłączenia magnetofonów. Podłączenie do komputera wykonujemy za pomocą odpowiedniego wtyku. Wtyk ten o rozstawie wyprowadzeń 4 mm najprawdopodobniej będziemy musieli wykonać we własnym zakresie (lub kupić na Zachodzie).

Dalsza rozbudowa

Problem rozbudowy magnetofonu o dalsze przystawki (elektroniczny licznik obrotów, oddzielny zasilacz, itp.) rozwiązać możemy zabudowując w magnetofonie złącza dostępne na naszym rynku i wyprowadzając na nie odpowiednie sygnały. Na rysunku obok tab. 2 przedstawiono uniwersalny układ łączówek. Aby go wykonać na tylnej ściance obudowy montujemy gniazdo 9-stykowe 881 009 (producentem jest Eltra), wtyk tego samego rodzaju 871 009 oraz gniazdo 25-stykowe 881 025. Pomiędzy złączami 9-stykowymi a gniazdem 25-stykowym wykonujemy odpowiednie połączenia. Następnie wylutowujemy z obwodu drukowanego magnetofonu oryginalny przewód połączeniowy i zakończamy gniazdem 9-stykowym z obudową (pierścienie ferrytowe nawleczone na kable bez trudu mieszczą się w obudowie gniazda). Z kolei podłączamy odpowiednie punkty układu magnetofonu (te, z których uprzednio wylutowaliśmy przewód połączeniowy) do gniazda 25-stykowego i wykonujemy odpowiedni zwieracz z wtyku 25-stykowego. W ten sposób, nie psując wyglądu magnetofonu przystosowaliśmy go do podłączenia wszelakiego rodzaju przystawek (także tej do kopiowania programów).

Styk nr	Adres	Przesyłany sygnał
1	M1 - [redacted]	M1 - [redacted]
2	M1 - +5V	M1 - +5V
3	M1 - +9V	M1 - +9V
4	M1 - odczyt	M1 - odczyt
5	M1 - zapis	M1 - zapis
6	M1 - czujnik	M1 - czujnik
7	NC	
8	NC	
9	NC	sygnały magn. M1
10	NC	bezpośrednio
11	NC	podane na Z3/G
12	NC	
13	NC	
14	Z2/G - 1	M2 - masa
15	Z2/G - 2	M2 - +5V
16	Z2/G - 3	M2 - +9V
17	Z2/G - 4	M2 - odczyt
18	Z2/G - 5	M2 - zapis
19	Z2/G - 6	M2 - czujnik
20	Z1/W - 1	K – [redacted]
21	Z1/W - 2	K – +5V
22	Z1/W - 3	K – +9V
23	Z1/W - 4	K – odczyt
24	Z1/W - 5	K – zapis
25	Z1/W - 6	K – czujnik

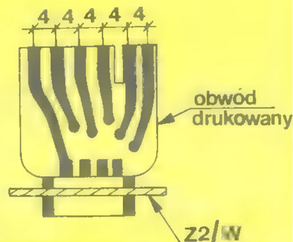
Tab. 6. Połączenie gniazda Z3/G mocowanego w magnetofonie do podłączenia przystawek, na które doprowadzone są wszystkie sygnały

Tab. 7. Połączenia wtyku Z3/W — zwory służącej do przywrócenia normalnej konfiguracji magnetofonu

Styk nr	Adres	Przesyłany sygnał
1	Z3/W - 20	M1-masa
2	Z3/W - 21	M1-+5V
3	Z3/W - 22	M1-+9V
4	Z3/W - 23	M1-odczyt
5	Z3/W - 24	M1-zapis
6	Z3/W - 25	M1-czujnik
7		
8		
9		sygnały mag. M1
10		bezpośrednio
11		podane [redacted] Z3/G
12		
13		
14		M2-masa
15		M2-+5V
16		M2-+9V
17		M2-odczyt
18		M2-zapis
19		M2-czujnik
20	Z3/W - 1	K -masa
21	Z3/W - 2	K -+5V
22	Z3/W - 3	K -+9V
23	Z3/W - 4	K -odczyt
24	Z3/W - 5	K -zapis
25	Z3/W - 6	K -czujnik

Tab. 5. Połączenia wtyku Z2/W zakończonego odpowiednim obwodem drukowanym, dającego możliwość podłączenia oryginalnej wtyczki magnetofonu M2

Styk nr	Adres	Przesyłany sygnał
1	M2 – A-1	A-1/M2 masa
2	M2 – B-2	B-2/M2 +5V
3	M2 – C-3	C-3/M2 +9V
4	M2 – D-4	D-4/M2 odczyt
5	M2 – E-5	E-5/M2 zapis
[redacted]	M2 – F-6	F-6/M2 czujnik
7	NC	sygnały magn. M2
[redacted]	NC	przez to złącze
9	NC	podawane są [redacted] Z3





GEOS

WIESŁAW SZYDŁOWSKI
Australia

Nowe możliwości C64

Często zdarza się tak, że stworzenie nowego oprogramowania dla starszego typu komputera potrafi w istotny sposób „odświeżyć" cechy użytkowe nie najnowszej już konstrukcji. Dobrym tego przykładem może być graficzny system operacyjny GEOS (ang. *Graphic Environment Operating System*) połączony z pakietem programów użytkowych porozumiewających się z użytkownikiem za pomocą joysticka lub myszki oraz ekranu wzbogaconego o elementy graficzne takie, jak np. piktogramy i okienka. W podobny sposób działają systemy operacyjne i programy użytkowe komputerów 16-bitowych Macintosh i Atari ST, na których zresztą wzorowali się twórcy GEOS-a. Dużym ograniczeniem dla nich były niewielkie możliwości sprzętowe C64 (8-bitowy mikroprocesor Mostek 6510 i 64 KB RAM), ale rezultaty przeszły chyba oczekiwania.

System operacyjny GEOS opracowany został przez firmę Berkeley Softworks z Kalifornii przy współpracy z Commodore Business Machines Inc. Firma Commodore ogłosiła GEOS oficjalnym systemem operacyjnym dla komputerów C64 i C128. Na razie jest to opcjonalny system — użytkownik może stosować Commodore DOS Kernal i w razie potrzeby uruchomić nowy system. Prace rozwojowe nad GEOS-em trwają nadal: dopiero ostateczna wersja będzie dostępna w pamięci ROM lub też w postaci wymiennego modułu (ang. *cartridge*). Z uwagi na częste używanie pamięci dyskowej VC 1541, która jak wiadomo jest stosunkowo wolna, zaszła potrzeba przyspieszenia operacji czytania i zapisu danych z tego urządzenia. Dzięki programowi diskTurbo linia używana do przesyłania sygnałów czasowych została wykorzystana dodatkowo do przesyłania danych i współpracuje asynchronicznie z główną linią transmisji danych, kilkakrotnie przyspieszając dostęp do pamięci zewnętrznej. Dalsze korzyści przynieść może zastosowanie dwóch jednostek pamięci dyskowej, z których jedna przydzielona jest dla dysku systemowego a druga dla dysku użytkownika.

Wprowadzony ostatnio na rynek moduł pamięci 1764 pozwala na rozszerzenie RAM-u C64 do 256 KB. Część tej pamięci może być wykorzystana przez GEOS jako tzw. dysk pamięciowy (ang. *RAM disc*). Niedawno na rynku w USA pojawiły się twarde dyski o dużej pojemności, które można podłączyć do Commodore 64 lub 128. Jeden z nich — ICT Data Chief o pojemności 20 MB jest zgodny z systemem operacyjnym GEOS. Technicznie możliwe jest podłączenie trzech takich jednostek, co daje łączną pojemność 60 MB (!). Dostęp do menu oraz komend systemu realizowany jest poprzez użycie joysticka lub myszki, których ruchy przesuwają kursor na ekranie a przyciśnięcie odpowiedniego przycisku (*fire*) aktywuje wybraną komendę. Tworzenie grafiki komputerowej może być sterowane także poprzez digitizer — tabliczkę Koala Pad lub pióro świetlne. GEOS pozwala na tworzenie grafiki oraz tekstów o rozdzielczości 80 punktów na cal. Rozmiar punktu wynosi 0,31 mm. Wprowadzenie danych na drukarkę dokonywane jest przez specjalny program sterujący. Do wyboru jest ponad 30 programów do różnych drukarek, między innymi do kolorowych Okimate 20 lub Canon JX80, jak również drukarek laserowych.

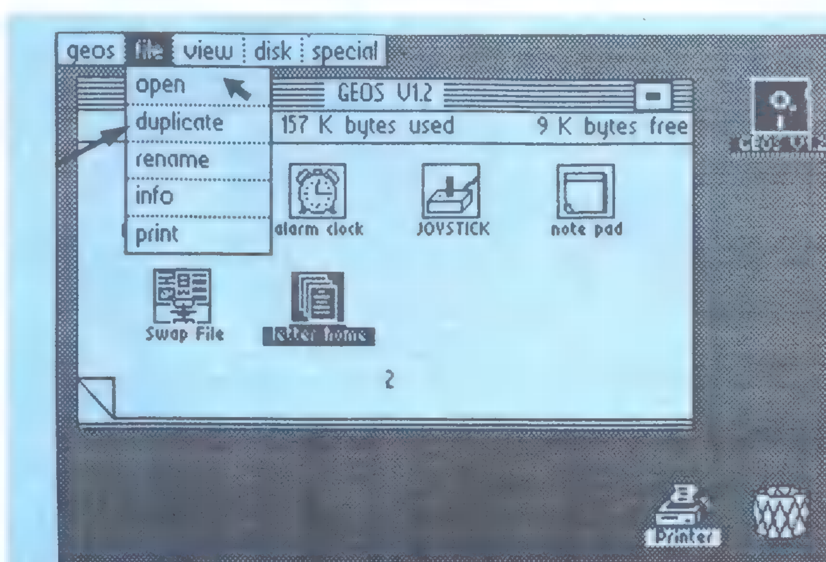
Pracując w systemie operacyjnym GEOS istnieje możliwość uruchamiania programów napisanych w języku BASIC. Jedynym limitem jest wielkość programu, która nie może przekraczać 26 KB. Jeżeli program jest ładowany poprzez GEOS korzysta on z diskTurbo, co skraca tę operację o ok. 10 razy. Użytkownik ma również możliwość wyjścia z systemu GEOS do systemu Commodore DOS. Powrót do GEOS następuje poprzez naciśnięcie klawiszy STOP/RESTORE przy włożonym oryginalnym dysku systemowym w stacji dysków.

System operacyjny

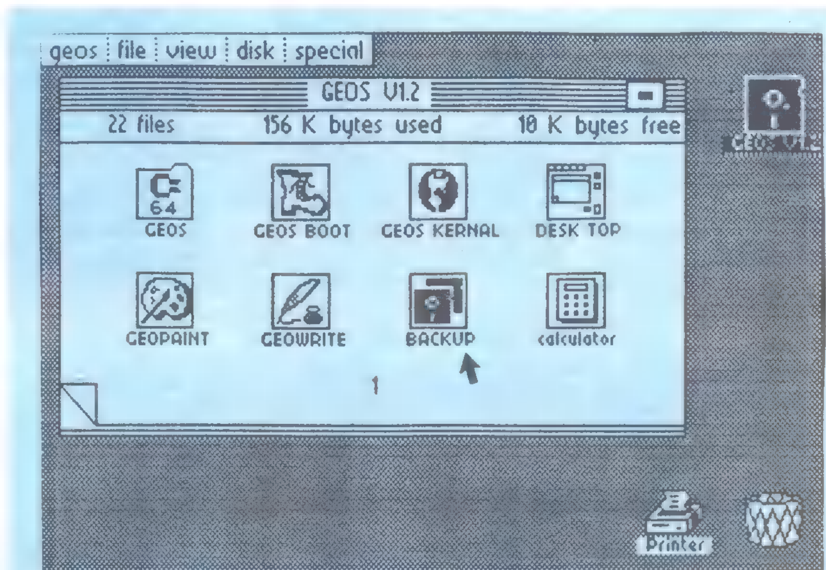
Po załadowaniu programu z oryginalnego dysku użytkownik musi zrobić kopię roboczą dysku systemowego, której będzie używać do swoich celów. GEOS posiada dość dobre zabezpieczenie przed kopiowaniem. Wykonana kopia jest „niepracująca". Uruchomienie systemu musi następować zawsze z oryginalnej dyskietki, którą potem usuwa się i pracuje na kopii roboczej. Ponieważ dysk systemowy wypełniony jest w pełni programami, nie ma na nim miejsca na zbiory użytkownika. Należy więc przygotować

kilka wersji roboczych dysków (najlepiej nadać każdemu inną nazwę), z których usunąć trzeba zbędne dla danego zastosowania programy. Można skasować takie programy, jak GEOS, GEOS KERNAL oraz GEOS BOOT. Na dyskietce pozostawia się programy kontrolujące wejście i wyjście danych oraz potrzebny program np. geoPaint. Kopię wykonuje się za pomocą programu użytkowego GEOS Backup. Można też użyć np. COPYQ.

Po uruchomieniu systemu na ekranie pojawia się tzw. deskTop. Jest to wejście do wszystkich programów systemu operacyjnego. Użytkownik może otwierać, zamykać, jak również ładować i zapisywać zbiory na dysk. Może również formatować dysk, usuwać zbędne zbiory lub też dokonywać weryfikacji zapisu. Wszystkie te operacje ułatwione są poprzez możliwość przeglądania zawartości dysku — nazwy zbiorów mogą być sortowane alfabetycznie, według wielkości, rodzaju lub też daty ostatniej zmiany. Poprzez deskTop można też wydrukować każdy zbiór na drukarce. GEOS deskTop zawiera główne menu, przez które następuje dostęp do poszczególnych komend. Na ekranie widoczne są również piktogramy reprezentujące jednostki pamięci dyskowej (maks. 2), drukarkę oraz „kosz na śmieci", który używany jest do kasowania zbiorów. DeskTop posiada wbudowany podręczny notatnik, który umożliwia dokonywanie krótkich notatek. Można zapisać w notatniku 127 stron, po 250 znaków na stronie. DeskTop zawiera również specjalny program kontrolny zwany Preference Manager, pozwalający na zmianę koloru tła, tekstu oraz koloru obrzeża ekranu. Można również zmienić kolor i kształt kursora, jak również jego prędkość w czasie przesuwania joysticka lub myszki. Wszystkie te zmiany mogą być zapamiętane na dyskietce roboczej. Dodatkowo deskTop posiada czterodziałaniowy kalkulator oraz wbudowany zegar z alarmem. Kalkulator można wyświetlić w okienku podczas np. pisania tekstu. Naciskanie „klawiszy" kalkulatora dokonywane jest przez przesunięcie kursora i naciśnięcie przycisku myszki lub joysticka. Wbudowany zegar działa poprawnie przy częstotliwości sieci 60 Hz, jaka jest stosowana w USA. Ustawiony czas wykorzystywany jest m.in. przy zapisie zbioru lub programu, kiedy oprócz nazwy zbioru zapisywane są również czas i data.



Tak wygląda deskTop systemu GEOS. U góry znajduje się główne menu, z którego użytkownik wybiera potrzebną opcję. Nazwa dyskietki umieszczona jest u góry — GEOS V1.1. Mały prostokąt z czarnym środkiem służy do zamykania zbioru. Informacja w następnym rzędzie dotyczy ilości zbiorów na dysku (22), objętości zajmowanej pamięci na dyskietce (156 KB) oraz wolnej pamięci (10 KB). Poniżej wyświetlane są piktogramy przedstawiające programy i zbiory znajdujące się na dysku. Jest to strona pierwsza, jak wskazuje numer 1 u dołu strony. Aby wyświetlić następne strony, należy ustawić kursor na „ośle ucho" — u dołu po lewej stronie i nacisnąć przycisk na joysticku. Dolny obszar służy do kopiowania zbiorów z jednej dyskietki na drugą — tutaj przesuwa się piktogram, symbolizujący dany zbiór. W prawym dolnym rogu wyświetlane są piktogramy symbolizujące drukarkę oraz „kosz na śmieci" czyli miejsce gdzie usuwa się niepotrzebne zbiory

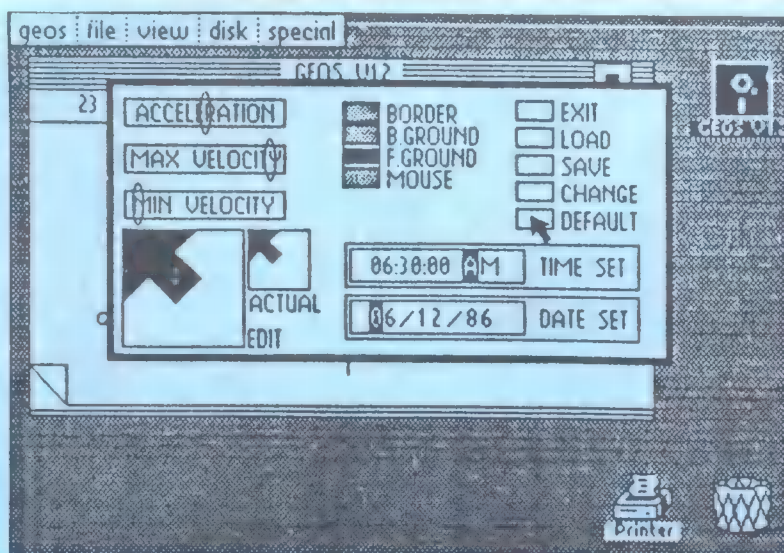


Przykład selekcji opcji z poziomu głównego menu

Edytor tekstowy geoWrite

Wersja GEOS 1.2 sprzedawana jest z dwoma programami użytkowymi: geoWrite — do tworzenia i obróbki tekstów oraz geoPaint — do tworzenia grafiki komputerowej. GeoWrite jest prostym edytorem tekstowym działającym według zasady: „what you see is what you get" — otrzymujesz to, co widzisz. Jego główną zaletą jest możliwość łatwej zmiany kroju pisma (ang. *font*) oraz łączenia tekstu z grafiką, która może być tworzona przez geoPaint. Transfer tekstu do geoPaint dokonywany jest przy pomocy specjalnego programu o nazwie Text Manager. Użytkownik ma do wyboru następujące kroje pisma: BSW, California, Cory, Dwinelle, Roma oraz University. Każdy z nich ma kilka wielkości czcionki — od 5 do 24 punktów. Można też wybrać kilka typów — jak pismo proste, kursywa (*italic*), grube (*bold*), podkreślone lub też wykreślone (*outline*). Te cechy geoWrite są szczególnie przydatne przy tworzeniu stron tytułowych, napisów, itp.

GeoWrite jest czasami dość „nieprzyjemny" w użyciu. Osobom, które dużo piszą zalecałbym raczej używanie takich programów jak Easyscript lub Speedscript, które mają bardziej rozbudowane komendy edycyjne. Szczególnie kłopotliwe jest przesuwanie strony, która nie jest wyświetlana w całości. Pomniejszoną kilkakrotnie



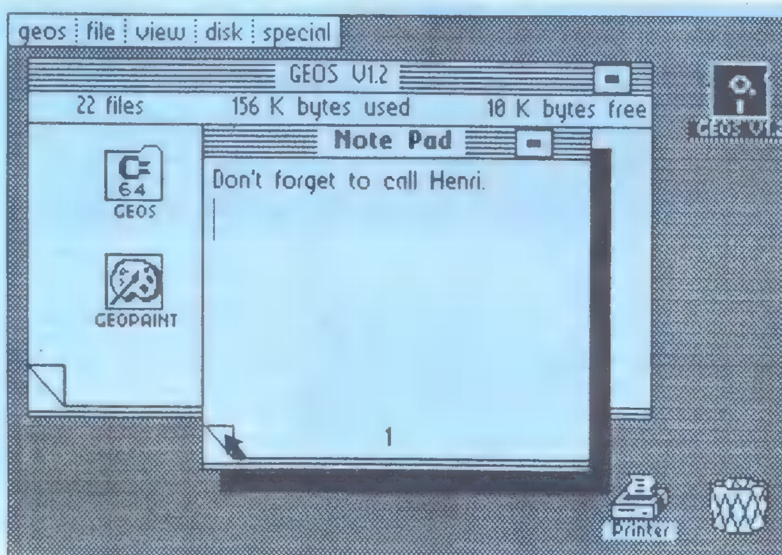
Tak wygląda ekran wyświetlany przez program kontrolny — preference manager. Użytkownik ustawia kolory ekranu, tekstu, obrzeża ekranu oraz kolor kursora. Ma też możliwość ustawienia parametrów kursora — jego wielkości oraz prędkości przesuwu. Można również ustawić czas oraz datę

stronę można wyświetlić w celu sprawdzenia rozłożenia tytułów, kolumn, itp. Te wszystkie bolączki i wady pierwszej wersji zostały usunięte w nowej wersji geoWrite 2.0 o nazwie Writer's Workshop. Jest to „prawdziwy" procesor tekstu, z wieloma udogodnieniami charakterystycznymi dla profesjonalnego oprogramowania tego gatunku. W porównaniu z poprzednią wersją, posiada możliwość uruchamiania komend z klawiatury, wykonywania nagłówków, selekcji stron, spacjowania oraz poszukiwania i wymiany wyrazów. Nowa wersja ma dodatkowy program użytkowy zwany Text Grabber. Pozwala on na czytanie oraz konwersję tekstów utworzonych przez takie edytory, jak EasyScript, SpeedScript oraz PaperClip.

Program graficzny geoPaint

Drugi program użytkowy — geoPaint zaprojektowany został do tworzenia grafiki komputerowej o dużej rozdzielczości drukowania tej grafiki na drukarce. Na formacie papieru A4 można tworzyć obrazy o rozdzielczości 640 na 800 punktów. Ponieważ nie można wyświetlić na ekranie całego formatu A4 w takiej rozdzielczości wyświetlana jest tylko część obrazu o rozmiarach 330 na 180 punktów. Resztę utworzonego obrazu przeglądać można poprzez użycie pamięci dyskowej jako pamięci wirtualnej, z której są odczytywane pozostałe jego części. Użytkownik ma możliwość wyświetlania całego obrazu, w kilkakrotnym zmniejszeniu, co przydatne jest dla oceny całości kompozycji. Taki zmniejszony obraz wyświetlany jest bez kolorów, w 16 stopniach szarości. GeoPaint posiada bogaty zestaw „narzędzi" pozwalających na tworzenie ciekawych efektów graficznych: różnego rodzaju „pędzle", powiększanie elementów obrazu (ang. *zoom*) i możliwość stosowania 16 kolorów. Można również łączyć grafikę z tekstem. Wybór odpowiednich „narzędzi" oraz kolorów następuje poprzez przesunięcie kursora na odpowiedni piktogram oraz przyciśnięcie guzika na myszce lub joysticku. Do wyboru jest 14 różnych narzędzi (pędzle, ołówek, gumka), 32 różne rozmiary „pędzli" oraz 32 wzory geometryczne, którymi można używać do wypełniania płaszczyzn (np. motyw ceglanego muru). Tworzenie szczegółów graficznych może odbywać się w tzw. *pixel mode* — trybie punktowym, kiedy to wyświetlane są poszczególne punkty obrazu w dużym powiększeniu. Ilość kolorów używanych jednocześnie ograniczona jest wtedy do 8. Transfer obrazów z geoPaint do geoWrite dokonywany jest przez program Photo Manager.

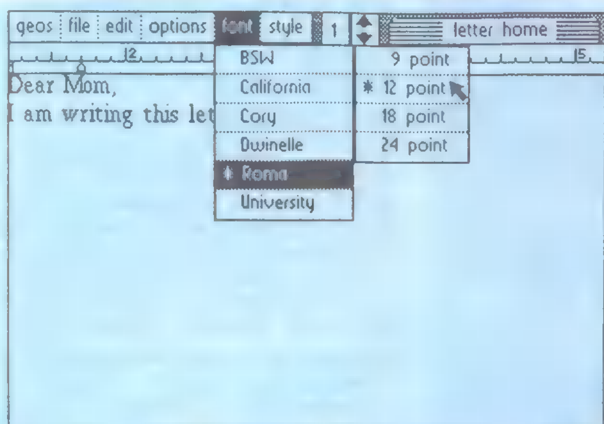
Przykład użycia notatnika podręcznego



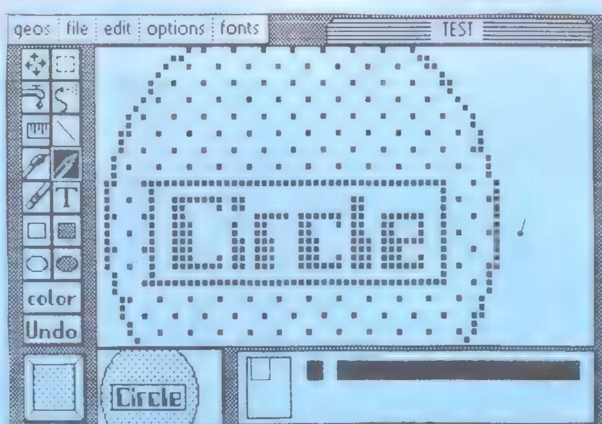
Co jeszcze potrafi GEOS?

Prace nad rozwojem systemu GEOS trwają nadal. Ostatnia wersja ma numer 1.3. Powstało też wiele programów użytkowych, które wykorzystują charakterystyczne cechy tego systemu. Program o nazwie fontPack1 zawiera komplet ponad 20 nowych wzorów pisma, w różnych kształtach i rozmiarach. Pozwala to na rozszerzenie repertuaru rodzajów pisma do około 30. Jedynym mankamentem jest możliwość korzystania jednocześnie tylko z siedmiu typów pisma. Zestaw deskPack1 zawiera trzy programy użytkowe oraz grę. Najważniejszy z nich to Graphic Grabber, który pozwala na transfer grafiki z takich programów, jak Print Shop, Newsroom oraz Print Master. Program Calendar zawiera kalendarz pozwalający na prowadzenie krótkich notatek dla każdego dnia dotyczących np. spotkań, narad czy uroczystości. Tworzenie własnych piktogramów umożliwia program Icon Editor. Dotychczas piktogramy symbolizujące zbiory lub programy nie będące w formacie zgodnym z systemem GEOS, były przedstawiane przez deskTop w postaci napisu „C64". Po zmianie formatu, która dokonywana jest w ciągu kilku sekund, użytkownik może zaprojektować własny piktogram. Zmianie podlega tylko format zapisu ścieżki indeksowej (numer 18). Po zmianie formatu dyskietka może być nadal używana bez przeszkód w systemie operacyjnym DOS.

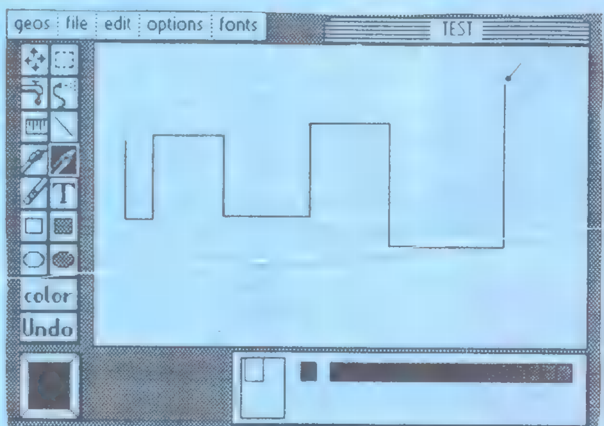
Nowym programem jest geoCalc. Jest to elektroniczny arkusz obliczeniowy, o pojemności 112 kolumn na 256 wierszy, co daje ponad 28 000 komórek. Użytkownik ma możliwość dzielenia ekranu na dwie części, w celu wyświetlenia interesujących go fragmentów arkusza. Obliczenia aryt-



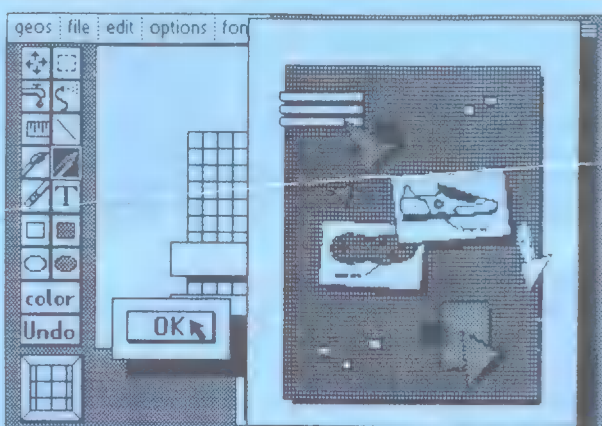
Tak wygląda ekran edytora tekstowego geoWrite. Przykład selekcji rodzaju pisma



Przykład użycia trybu punktowego (ang. *pixel mode*). W okienku na dole strony widoczny jest element w takiej wielkości, w jakiej zostanie włączony do rysunku



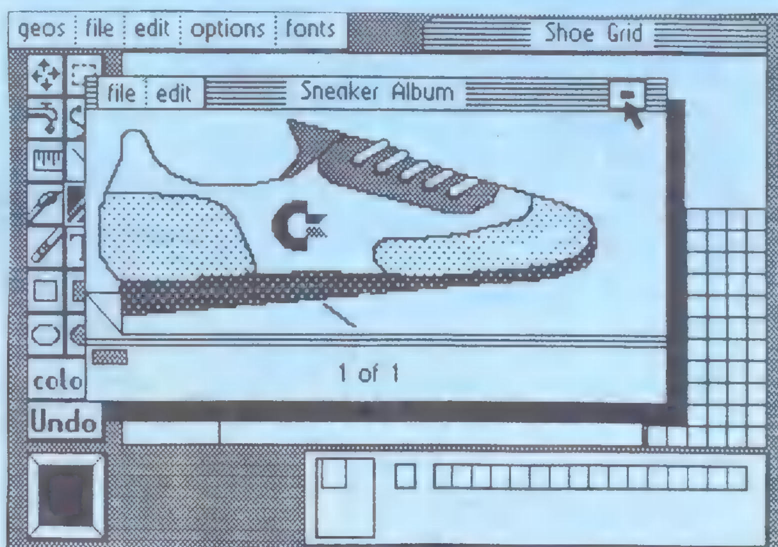
Ekran programu graficznego geoPaint. Z lewej zestaw „narzędzi" malarskich. U dołu, pośrodku ekranu pokazana jest cała strona z zaznaczeniem prostokąta będącego obecnie przedmiotem pracy



Wyświetlenie całej strony w kilkakrotnym zmniejszeniu dla oceny całości kompozycji przed jej wydrukowaniem

metyczne wykonywane są z dokładnością do 9 miejsc po przecinku. Program o nazwie geoDex służy do tworzenia własnych zbiorów danych (np. listy nazwisk i adresów znajomych). Dodatkowy program geoMerge zawarty na tej dyskietce pozwala na użycie tego zbioru do drukowania nalepek z adresami na listy lub włączanie adresów i nazwisk do listów napisanych przy użyciu edytora geoWrite. Do założenia i obsługi dużej ilości danych służy program bazy danych geoFile. Pozwala on na edycję, sortowanie oraz wyszukiwanie informacji zgodnie z wymaganiami użytkownika. Wspomniany już program Writer's Workshop posiada możliwość tworzenia wydruków na drukarce laserowej. Oczywiście przeciętny użytkownik nie może marzyć o zakupie sprzętu za ok. 2000 dolarów. Funkcja ta może być wykorzystana poprzez użycie drukarek podłączonych do amerykańskiej sieci komputerowej o nazwie Quantum Link. Na drugiej stronie oryginalnej dyskietki umieszczony jest program telekomunikacyjny pozwalający na dostęp do tej sieci.

Element kompozycji (but) w naturalnej wielkości

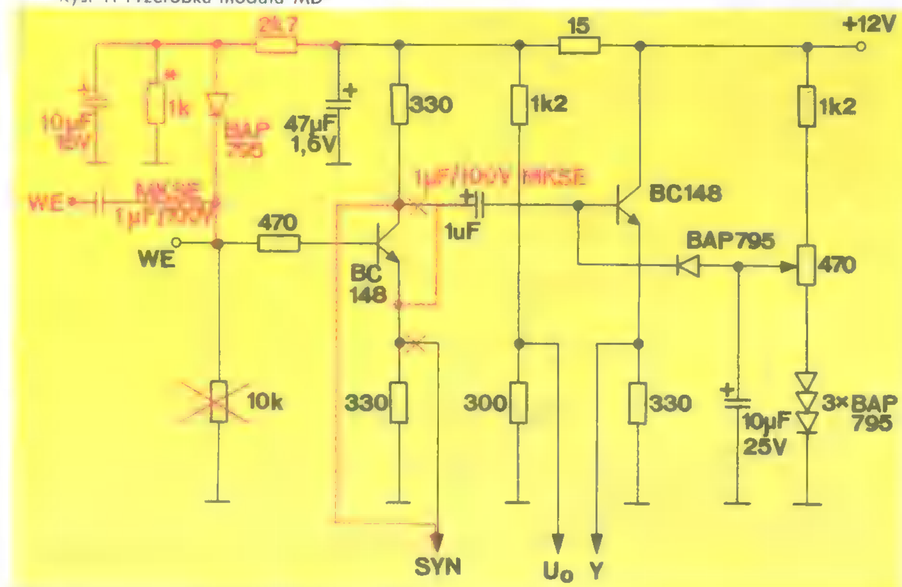




Jak przerobić telewizor na monitor komputerowy

Nieco ponad rok temu w „Młodym Techniku" 9/86 opisaliśmy konstrukcję monitora wykorzystującego moduły i podzespoły Neptuna 150, zaś w „Infor-Miku" 3/87 przeróbkę Veli 202 na monitor. **W tym miejscu chciałbym przeprosić wszystkich zainteresowanych za błąd w tytule, za który również ja jestem odpowiedzialny** — otóż przeróbka ta dotyczy starszego modelu telewizora Vela 202, a nie modelu Vela 203, jak wymieniono w tytule. W imieniu swoim i redakcji dziękuję za zasygnalizowanie nam owej pomyłki. Poniżej opisane będą sposoby przerobienia na monitor komputerowy najczęściej spotykanych w Polsce małych telewizorów czarno-białych, w tym również Veli 203.

Rys. 1. Przeróbka modułu MD



Przeróbka modułu MD

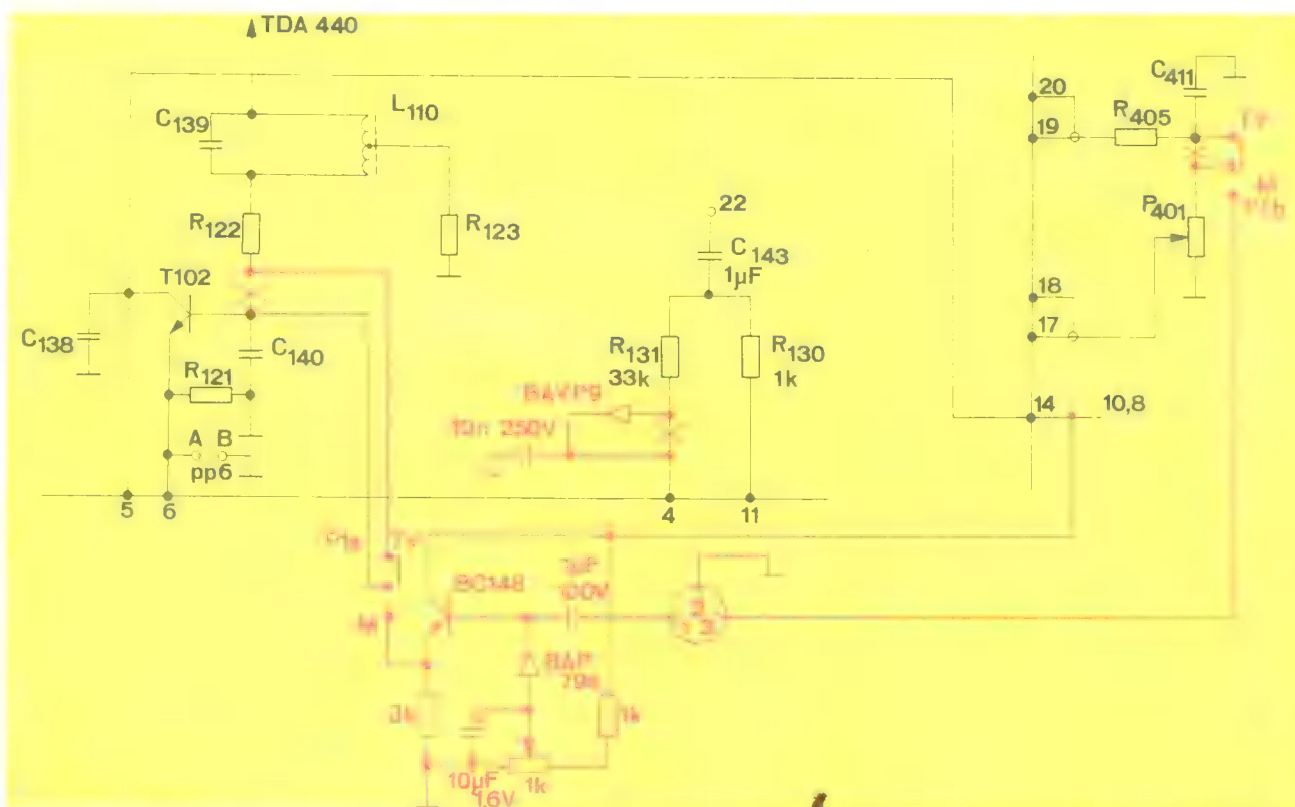
Najpierw opiszemy przeróbkę modułu MD monitora z numeru „MT" 9/86 tak, aby jego wejście odpowiadało standardowemu sygnałowi wideo komputera. Dla uproszczenia konstrukcji tej (następnych również) przyjmiemy jednostronne dopasowanie przewodu, pozwalające na uzyskanie dwukrotnie większej amplitudy sygnału wizyjnego, bez zauważalnego pogorszenia jakości obrazu. Schemat modułu MD z zaznaczonymi czerwonym kolorem zmianami przedstawia rys. 1. Najważniejszą zmianą jest zamiana podłączenia pierwszego stopnia z tranzystorem BC148. W wersji dla ZX Spectrum pracował on dla sygnału wizyjnego jako odwracacz fazy. Obecnie ma on spełniać rolę wtórnika. Sygnał o odwróconej fazie wykorzystywany jest do sterowania modułu synchronizacji. Dodatkowo na wejściu wprowadzono sprzężenie zmiennoprądowe, które uniezależnia nas od wartości składowej stałej na wyjściu wideo komputera. Układ polaryzacji bazy pierwszego tranzystora (także elementy dodatkowe) utrzymuje na stałym poziomie minimalną wartość napięcia na jego emiterze, zapewniając tym samym optymalne warunki pracy tego stopnia, niezmienne przy zmianach treści obrazu. Pozostałe elementy modułu oraz sposób regulacji pozostają nie zmienione w stosunku do poprzedniej wersji.

Nie podajemy rysunku płytki drukowanej nowej wersji modułu — dodatkowe elementy można z powodzeniem zamontować „w powietrzu", bez konieczności mozolnego przerabiania płytki.

Przeróbka Veli 203

Na rys. 2 przedstawiony jest schemat przeróbki Veli 203 na monitor — układy dodatkowe są niemal identyczne, jak w przypadku Veli 202. Różnice wynikają z odwrotnej polaryzacji zasilania (względem masy) oraz zastosowania w układzie p.cz. układu scalonego TDA 440. A zatem identycznie dołączamy wejście fonii (do potencjometru regulacji siły dźwięku), tak samo włączamy diodę oraz kondensator 10 nF w obwodzie regulacji jaskrawości (poprawia to stałość podtrzymania poziomu czerni). W układzie wejściowym zamiast stałego dzielnika zastosowano potencjometr montażowy. Przy uruchamianiu podajemy na wejście monitorowe obraz z komputera o średnim poziomie bieli. Może to być np. czarno-biała szachownica lub sygnał pasów kolorowych. Regulujemy potencjometrem tak, aby na emiterze T102 napięcie było możliwie bliskie wartości uzyskiwanej przy normalnym odbiorze programu telewizyjnego.

Przy montażu należy zwrócić szczególną uwagę na długość przewodów prowadzących sygnały wizyjne — zbyt długie



Rys. 2. Podłączenie wejścia monitorowego do Veli 203

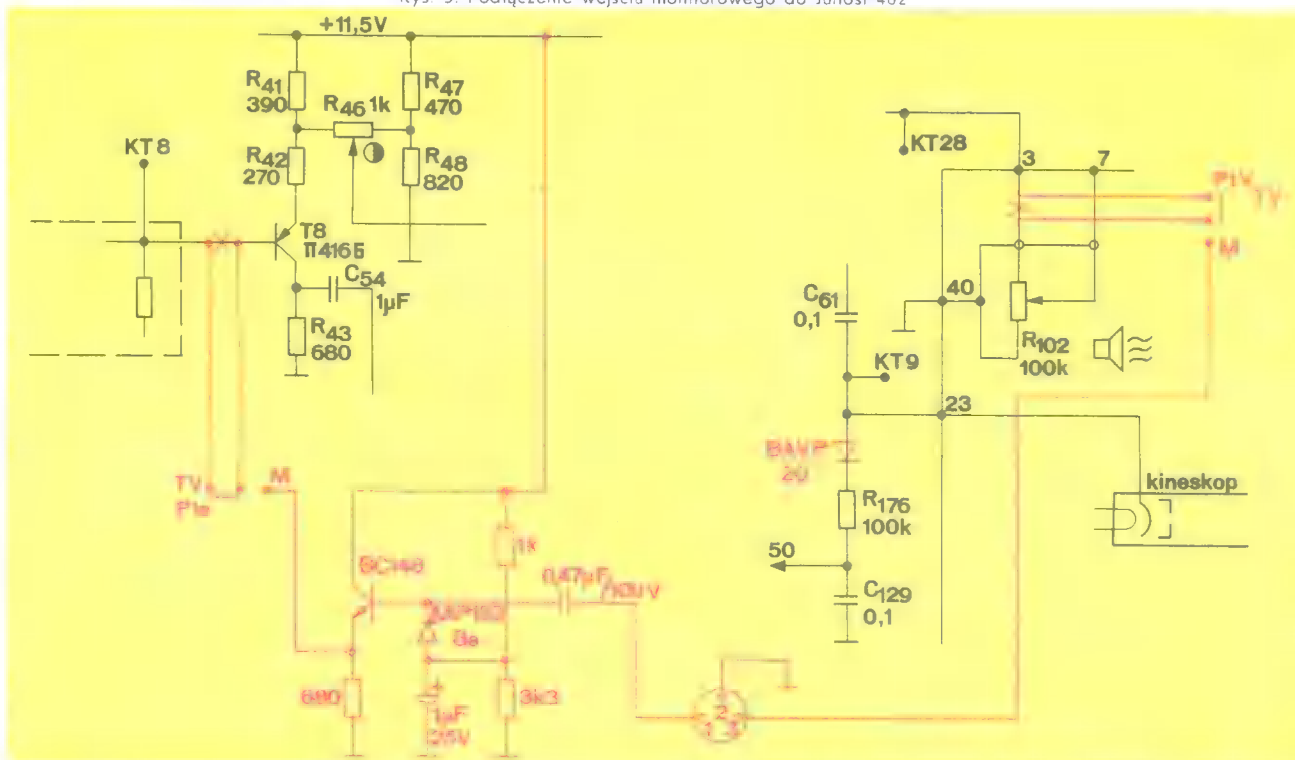
połączenia względnie niewłaściwe ich prowadzenie może wpłynąć na znaczne pogorszenie jakości obrazu. Dla połączeń sygnałów akustycznych zaleca się stosowanie przewodów ekranowanych. Przełącznik może być dowolnego typu, najlepiej pojedynczy isostat niezależny.

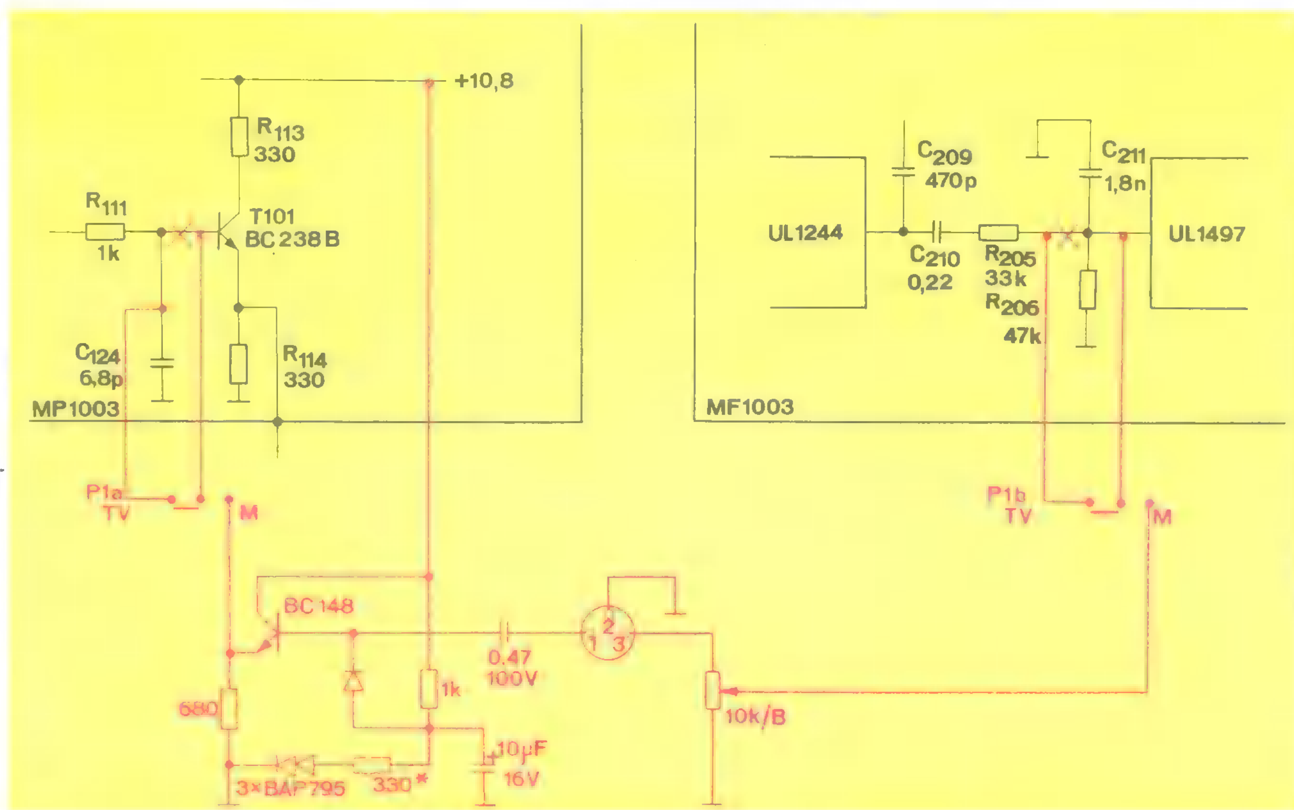
Podłączenia do odpowiednich miejsc płytki najłatwiej dokonać przez przecięcie ścieżek i dolutowanie przewodów. Należy przy tym bardzo dokładnie określić miejsce przecięcia ścieżki, każdorazowo kontrolując połączenia z elementami po stronie montażu ze schematem ideowym.

Przeróbka Junost 402

Na rys. 3 przedstawiona jest przeróbka popularnego telewizora turystycznego produkcji radzieckiej Junost 402. Układ dopasowujący jest niemal identyczny, jak w poprzedniej wersji — różnice dotyczą wartości elementów oraz zastosowania

Rys. 3. Podłączenie wejścia monitorowego do Junost 402





Rys. 4. Przeróbka Neptuna 150 na monitor

diody germanowej zamiast krzemowej (nie jest to wymóg konieczny). Cechą charakterystyczną zastosowanych układów dopasowujących jest odtwarzanie składowej stałej na poziomie impulsu synchronizacji, który jest praktycznie niezależny od treści obrazu. Uzyskujemy dzięki temu względnie dobrą stabilizację poziomu czerni na ekranie, oczywiście, o ile wzmacniacz wizji ma sprzężenie stałoprądowe. Ponieważ Junost (podobnie jak Vela) ma sprzężenie zmiennoprądowe i tutaj stosujemy w układzie regulacji jaskrawości (obwód katody kineskopu) diodę częściowo odtwarzającą składową stałą — wizualnie daje to dość znaczną poprawę.

Podłączenie wykonujemy podobnie, jak w poprzednich układach, identyczne są też wskazówki montażowe. Przy zastosowaniu sprawnych elementów nie jest konieczna jakakolwiek regulacja czy dobór elementów.

Przeróbka Neptuna 150

W przypadku Neptuna 150 przeróbka również nie jest zbyt skomplikowana (rys. 4). Układ odtwarzający składową stałą jest niemal identyczny jak w poprzednich wersjach, zastosowano jedynie elementy o stałych wartościach oraz dwie diody krzemowe poprawiające nieco stabilność termiczną układu (w Neptunie zastosowane jest sprzężenie stałoprądowe, a zatem stabilność poziomu czerni jest dość istotna — w razie konieczności można korygować poziom czerni przy pracy monitorowej przez zmianę wartości rezystora oznaczonego gwiazdką). Pewien kłopot występuje jednak przy dołączeniu sygnału fonii. Ze względu na fakt, że w module MF1003 wykorzystano elektroniczną regulację siły głosu, konieczne jest wprowadzenie dodatkowego potencjometru regulującego poziom dźwięku dla pracy monitorowej (potencjometr wykładniczy 10 kΩ).

Montaż dodatkowych układów w przypadku Neptuna jest o tyle łatwiejszy, że poszczególne moduły można łatwo wyjąć i operacji przecinania ścieżek dokonujemy na module zdemontowanym. I tu nie należy zapomnieć o minimalizacji długości połączeń.

Kilka uwag praktycznych

W wielu listach zwracano nam uwagę na brak w monitorze z numeru „MT" 9/86 modułu dźwięku. Jego wykonanie oraz podłączenie nie powinno jednak być nawet dla niezbyt zaawansowanego radioamatora większym problemem — wykorzystujemy bądź standardowe układy wzmacniaczy m.cz., bądź też samodzielnie zbudowany najprostszy wzmacniacz o mocy rzędu 1—3 W, najlepiej na układzie scalonym.

Opisane powyżej układy są w zasadzie stosunkowo proste i teoretycznie ich wykonania mógłby się podjąć nawet niezaawansowany elektronik — amator. Należy jednak bardzo uważać przy montażu, gdyż ewentualna pomyłka spowodować może uszkodzenie drogiego odbiornika telewizyjnego. **Ponadto zasadą powinno być operowanie na układzie telewizora przy wyłączonym jego zasilaniu.** W przypadku trudności z uruchomieniem układów pośredniczących wskazane jest skorzystanie z oscyloskopu i porównanie standardowego sygnału wizyjnego przy odbiorze programu TV z sygnałem z układu dopasowującego i ewentualnie odpowiednia regulacja lub korekcja wartości elementów.

Uprzedzając ewentualne zapytania Czytelników dotyczące przeróbek dużych lampowo-tranzystorowych odbiorników TV (np. Libra, Saturn i pochodne) na monitory informuję, że jest to bardzo trudne — wynika to z trudności zapewnienia odpowiedniej izolacji między wejściem monitorowym (połączonym przecież z komputerem) a układem telewizora, połączonym galwanicznie z siecią! Konieczne byłoby zastosowanie izolacji najlepiej optoelektronicznej, lecz uzyskanie pasma rzędu 5 MHz lub więcej na krajowych elementach jest raczej niemożliwe. Zastosowanie innych rodzajów izolacji (np. modulacja i separacja za pomocą odpowiedniego transformatora) jest bardzo trudne, a uzyskane efekty mogą być gorsze od tych, które obserwujemy przy wykorzystaniu standardowego modulatora TV w komputerze.

Chciałbym jednocześnie podziękować Koledze Dariuszowi Łuszczakowi za sugestie dotyczące wersji modułu MD przystosowanej do komputera o znormalizowanym wyjściu wideo.

GRZEGORZ ZALOT

SEMINARIUM „INFORMIKA"

ASEMBLER — GENS 3

Część 5

Na wstępie kilka uwag o kodzie asemblera Gens 3. Wyjaśnijmy dokładnie ten problem, który wzbudził pewne wątpliwości u kilku dociekliwych Czytelników, a nie jest wyjaśniony dokładnie w popularnych opracowaniach. Jest on bezpośrednio związany z relokowalnością Gensa 3 i dotyczy również jego nowszej wersji Gens 3M2, a także wchodzących w skład oprogramowania podstawowego monitora Mons 3 i jego nowszej wersji — Mons 3M2. Wszystkie wymienione programy są relokowalne i zostały utworzone w podstawowej wersji z lokacją ORG = 0. W takiej postaci są one ładowane do pamięci operacyjnej pod wybrany adres. Oczywiście w przypadku tak złożonych programów konieczne jest stosowanie budowy strukturalnej (użycie procedur), co w przypadku odwołań do procedur zawartych wewnątrz kodu wymaga odwołań do adresów bezwzględnych. Drugim powodem odwoływania się do adresów bezwzględnych jest wykorzystanie zmiennych zawartych w obszarze działania programu. Widać stąd, że w przypadku każdej konkretnej lokacji programu adresy te powinny zostać odpowiednio zmienione, jeśli program ma działać poprawnie. Łatwo zauważyć, że do takich poprawek najlepiej nadaje się wersja programu z lokacją równą zeru. W tym przypadku sformułowanie „odpowiednio zmienione" znaczy po prostu „zwiększone o wartość aktualnej wybranej lokacji programu". Kody wymienionych programów składają się funkcjonalnie z trzech części: procedury dokonującej relokacji (umieszczonej na początku kodu), właściwego kodu programu oraz umieszczonej na końcu „tablicy" poprawek. Wymienione programy używają identycznej, liczącej 29 bajtów, procedury relokacyjnej. Długości wszystkich obszarów dla wymienionych programów przedstawia poniższa tabela:

TABELA

Program	Procedura	Kod	Tablica	Razem
Gens 3	29	6991	1334	8354
Gens 3M2	29	8403	1582	10 014
Mons 3	29	4809	922	5760
Mons 3M2	29	5057	982	6068

Po pierwszym uruchomieniu programów (tzw. zimny start) obszar tablicy nie jest wykorzystany i może zostać skasowany.

Opis błędów asemblacji

NUMER BŁĘDU	OPIS
1.	Błąd składni
2.	Słownik asemblera nie zawiera takiego mnemonika
3.	Nieprawidłowo sformułowany rozkaz
4.	Użyta nazwa już istnieje w tablicy symboli
5.	Linia programu zawiera znak nieważny w tym kontekście
6.	Jeden z argumentów rozkazu jest nieprawidłowy
7.	Nazwa użyta przez programistę jest nazwą zastrzeżoną
8.	Niepoprawne zestawienie rejestrów w linii programu
9.	Za dużo rejestrów w linii programu
10.	Wartość wyrażenia użytego w tym rozkazie jest za duża
11.	Rozkazy **JP IX+n** oraz **JP IY+n** nie istnieją
12.	Błędnie sformułowana dyrektywa asemblera
13.	Nazwa będąca argumentem dyrektywy EQU nie została wcześniej zdefiniowana
14.	Wykryto próbę dzielenia przez wyrażenie równe zeru
15.	Wykryto nadmiar w operacji mnożenia

Dodatkowe błędy wykrywane przez Gens 3M2

16.	Makroinstrukcja definiowana wewnątrz innej makroinstrukcji
17.	Błędny identyfikator definicji makroinstrukcji
18.	Wywołanie makroinstrukcji wewnątrz definicji makroinstrukcji
19.	Zagnieżdżone wyrażenie warunkowe

Uwagi:

ad 4. Błąd wystąpi przy próbie nadania danej nazwie nowej wartości, np. użycia dwukrotnie identycznej etykiety

ad 7. Błąd wystąpi w wypadku użycia nazwy zastrzeżonej w niewłaściwym miejscu np. w polu etykiety

ad 10. Błąd ten nie zatrzymuje procesu asemblacji w pierwszej fazie. Jeśli wymagana wartość wyrażenia przekracza 255 asembler wpisuje wartość modulo 256 i kontynuuje generowanie kodu. Ponieważ błąd ten jest sygnalizowany najczęściej w przypadku skoku względnego na zbyt dużą odległość, należy wykrycie tego błędu traktować jako konieczność zmiany w linii programowej rozkazu JR na JP.

Ponieważ opis błędów przedstawiony powyżej, wygląda zbyt enigmatycznie, zilustrujemy go wydrukiem programu źródłowego w Gens 3, którego jedyną zaletą jest popełnienie wszelkich możliwych błędów. Znalezienie przykładu ilustrującego błąd nr 3 pozostawiamy jako ćwiczenie cierpliwemu Czytelnikowi. Błędy 16—19 zostaną omówione dokładniej w dalszej części wykładu.

```
 1 START   LD    1,A
 2         LDA
 3 ;
 4 START   LD    A,1
 5         LD    A:1
 6         LD    A+1,1
 7 A       LD    A,1
 8         LD    IX,IY
 9         LD    (IX+A),B
10         LD    A,300
11         JP    (IX+1)
12         ORG
13 DLUG    EQU   KON-START
14         LD    HL,START/0
15 KON     LD    HL,2*30000

19 ;
20 ;blad w linii 10 wystapi
21 ;po usunieciu pozostalych
22 ;linii programu
```

Asembler prosty i makroasembler

Omówione do tej pory elementy asemblera Gens 3, a w szczególności możliwość używania nazw mnemonicznych rozkazów, adresów symbolicznych, wyrażeń arytmetycznych, pseudoinstrukcji i dyrektyw są charakterystyczne dla każdego języka symbolicznego. Praktyka programowania wykazała, że celowe jest wyposażenie asemblera w kilka dodatkowych mechanizmów, które usprawniają proces programowania. Należą do nich asemblacja warunkowa i makroinstrukcja. Pierwsza jest dostępna w Gens 3, a obie możliwości są w pełni dostępne w Gens 3M2. Zauważmy, że z dotychczas poznanych wiadomości wynika, że każda linia programowa jest analizowana i realizowana przez asembler w sposób bezwarunkowy. Ponadto każdej instrukcji programowej odpowiada dokładnie jeden rozkaz maszynowy mikroprocesora. Takie rozwiązanie nie jest jedyne. Rezygnacja ze spełnienia pierwszego warunku prowadzi do pojęcia asemblacji warunkowej, a niespełnienie drugiego do makroinstrukcji. Składnia pseudoinstrukcji asemblacji warunkowej to: **IF** [wyrażenie arytmetyczne] [ciąg linii programu] **ELSE** [ciąg linii programu] **END.** Nazwy **IF**, **ELSE**, **END** umieszczamy w polu operacji, zaś wyrażenie arytmetyczne w polu argumentów. Działanie pseudoinstrukcji jest następujące: jeżeli wartość wyrażenia arytmetycznego jest różna od zera to asemblowane są linie programu zawarte między **IF** i **ELSE**, a opuszczane są linie zawarte między **ELSE** i **END**. Jeżeli wartość wyrażenia arytmetycznego jest równa zero, asembler podejmuje odwrotne działanie. Nazwy **ELSE** można użyć wielokrotnie dla przedzielenia kilku grup linii programowych, bowiem w istocie każde jej użycie jest

równoznaczne zmianie aktualnego stanu asemblacji. (Przez stan asemblacji rozumiemy tu analizowanie lub pomijanie linii programowych). Można opuścić nazwę ELSE, jeżeli ciąg linii programowych po niej następujący jest pusty.

Zastosowania asemblacji warunkowej są dość różnorodne. Chociaż najbardziej spektakularne jest użycie jej w definiowaniu makroinstrukcji dostępnych w Gens 3M2, to użycie jej w programie napisanym w Gens 3 może przynieść duże korzyści. Częstym przypadkiem jest konieczność dołączania do testowanego programu, w kilku krytycznych miejscach, dodatkowych procedur umożliwiających ustalenie aktualnych wartości zmiennych. W gotowym programie zostają one usunięte z tekstu programu źródłowego, co nie jest najlepszym rozwiązaniem, jeśli w przyszłości program będzie podlegał modyfikacjom. Konieczność posiadania dwu, a często większej liczby wersji programu nie jest rozwiązaniem praktycznym. Innym przykładem może być konieczność posiadania kilku wersji programu różniących się niewielkimi zmianami parametrów niezbędnymi dla poprawnego działania w konkretnej konfiguracji. Korzyść wynikająca z zastosowania asemblacji warunkowej jest w tym przypadku podwójna. Uzależnienie parametrów od warunków pozwala dodatkowo sprecyzować ogólniejszy problem i udokumentować go w programie źródłowym.

Współpraca asemblera Gens 3 ze stacją dysków

Powszechnie ugruntowana w literaturze opinia o niemożności wykorzystania przeważającej liczby wartościowych programów przeznaczonych dla ZX Spectrum w przypadku wykorzystania innej pamięci zewnętrznej niż taśmowa, skłoniła autora do przedstawienia prostego programu umożliwiającego realizację podstawowych funkcji obsługi stacji dysków. Jak wiadomo producent przewidział wykorzystanie w charakterze szybkiej pamięci masowej tzw. *microdrive* i możliwość ich wykorzystania zapewnia asembler Gens 3M2. Niestety ten rodzaj pamięci nie rozpowszechnił się na większą skalę. Nie wnikając w analizę rzekomych i faktycznych wad tego urządzenia (autor eksploatuje je z powodzeniem od 3 lat!) stan faktyczny jest taki, że liczą się praktycznie na naszym rynku stacje dysków niezależnych producentów. Ze względu na ich popularność i dostępność została wybrana dla przykładu stacja dysków 3-calowych FDD-3 opisywana w numerze 1/86 „InforMika" przez Rolanda Wacławka. Stacja ta posiada bardzo dobry system operacyjny wykorzystujący większość zleceń, składnię i instrukcje właściwe dla interpretera BASIC-a wbudowanego w ZX Spectrum. Dodatkowym, istotnym dla przenoszenia oprogramowania czynnikiem, jest niewykorzystywanie przez system operacyjny pamięci operacyjnej mikrokomputera. Na stację tę można przenieść wiele programów użytkowych dokonując nieskomplikowanych adaptacji na poziomie BASIC-a.

W przedstawionym programie ograniczono się do niezbędnego minimum ze względu na obszerność wydruku, pozostawiając rozbudowę programu o dalsze zlecenia faktycznym potrzebom użytkownika. Za minimum przyjęto: możliwość wpisania, dopisania i zapisania programu źródłowego oraz zapisania programu wynikowego. Dla zrealizowania tych zadań najważniejszym problemem jest istnienie odpowiednich zmiennych wskazujących adresy odpowiednich obszarów asemblera. Jak się okazuje wszystkie niezbędne zmienne znajdują się w kodzie programu. Adresy tych zmiennych dla lokacji Gens 3 równej 30 000 znajdzie Czytelnik w liniach 9400—9430 wydruku. Pierwsza liczba określa zmienną wskazującą adres początku danego obszaru, druga adres końca danego obszaru (zmienne są oczywiście dwubajtowe). Ponadto przez adres końca należy rozumieć, zgodnie z przyjętą w asemblerze Gens 3 konwencją, adres pierwszej wolnej komórki pamięci po danym obszarze (patrz zlecenie X asemblera). Tak więc odjęcie adresu początku od adresu końca daje bezpośrednio w wyniku długość danego obszaru. Dla ułatwienia identyfikacji zbiorów na dysku przyjęto automatyczne dopisywanie do maksymalnie 8-znakowej nazwy programów źródłowych i wynikowych dodatkowych identyfikatorów, odpowiednio ASM i BIN. Nazwa programu źródłowego do wpisania lub dopisania i jego długość jest pobierana dla wygody bezpośrednio z ekranu po wykonaniu zlecenia katalogu, ponadto długość odczytana z ekranu jest pomniejszana o 5, ponieważ pięć pierwszych bajtów w pliku przeznaczonych jest na identyfikator wykorzystywany przez system operacyjny. Ponieważ asemblacja programu z generowaniem kodu wynikowego pod jego docelową lokacją nie jest w poważnych zastosowaniach zbyt uniwersalna, celowo założono wykonywanie asemblacji z wykorzystaniem opcji nr 16 (kod programu wynikowego jest wtedy umieszczany bezpośrednio za tablicą symboli). Jeśli nie wykorzystamy tej opcji asemblacja jest oczywiście możliwa. Nieprawidłowo będą jedynie wyświetlane po powrocie do BASIC-a lokacje dwóch ostatnich obszarów pamięci.

TADEUSZ BASISTA

```
   1 REM PROGRAM OBSLUGI GENS3
   2 REM DLA STACJI DYSKOW FDD3
   3 REM ZX SPECTRUM 48K
   4 REM (C) T. Basista 1987
   9
  10 DEF FN P(X)=PEEK X+256*PEEK
 (X+1)
 100 REM MENU
 110 CLS : GO SUB 9100
 120 GO SUB 200
 180 CLS : GO SUB M*1000
 190 GO TO 100
 199
 200 REM WYBOR KLAWISZA
 210 LET I$=INKEY$: IF I$="" THE
N GO TO 210
 220 LET M=CODE (I$)-48
 230 IF M<1 OR M>6 THEN GO TO 21
0
 240 RETURN
 249
 250 REM NAZWA Z KLAWIATURY
 260 INPUT "NAZWA: "; LINE N$
 270 LET N$=N$+"."+C$
 280 RETURN
 299
 300 REM NAZWA Z EKRANU
 305 PRINT #1;AT 0,0; INVERSE 1;
"KURSORY- ZMIANA  ENTER-WYBOR";
 310 LET W=6: LET I=0
 315 PRINT AT W,0; OVER 1; FLASH
 1;TAB 20;
 320 LET I=CODE INKEY$
 325 IF I THEN PRINT AT W,0; OVE
R 1; FLASH 0;TAB 20;
 330 LET W=W+(I=10 AND W<21)-(I=
11 AND W>6)
 335 IF I<>13 THEN GO TO 315
 340 LET M$=""
 345 FOR I=0 TO 19
 350 LET S$=SCREEN$ (W,I)
 355 LET M$=M$+S$
 360 NEXT I
 365 LET C$=M$(10 TO 12)
 370 IF C$<>"ASM" THEN GO TO 300
 375 LET N$=M$(1 TO 8)+"."+C$
 380 LET D$=M$(16 TO 20)
 385 LET DL=VAL (D$)-5
 390 RETURN
 399
 500 REM LOAD
 510 LOAD *N$CODE ST,DL
 520 RETURN
 599
 600 REM SAVE
 610 SAVE *N$CODE ST,DL
 620 RETURN
 699
 900 REM ADRES KONCA
 910 LET KON=ST+DL
 920 LET K2=INT (KON/256)
 930 LET K1=KON-256*K2
 940 POKE 30054,K1
 950 POKE 30055,K2
 960 RETURN
1000 REM WCZYTANIE ZRODLOWEGO
1010 LET ST=FN P(36679)
1020 CAT *"+.ASM": GO SUB 300
1030 GO SUB 500: GO SUB 900
1900 RETURN
1999
2000 REM DOPISANIE ZRODLOWEGO
2010 LET ST=FN P(30054)
2020 GO TO 1020
2999
3000 REM NAGRANIE ZRODLOWEGO
3010 LET ST=FN P(36679)
3020 LET KON=FN P(30054)
3030 LET DL=KON-ST
3040 LET C$="ASM"
3050 GO SUB 250: GO SUB 600
3900 RETURN
3999
4000 REM ZAPISANIE WYNIKOWEGO
4010 LET ST=FN P(36711)
4020 LET KON=FN P(36700)
4030 LET DL=KON-ST
4040 LET C$="BIN"
4070 GO SUB 250: GO SUB 600
4900 RETURN
4999
5000 REM POWROT DO ZRODLOWEGO
5010 RANDOMIZE USR 30061
5900 RETURN
5999
6000 REM POWROT Z KASOWANIEM
6010 RANDOMIZE USR 30056
6900 RETURN
6999
9100 REM WYDRUK MENU
9110 RESTORE : PRINT TAB 13;"MEN
U"
9120 FOR I=1 TO 6
9130 READ M$: PRINT AT 2+I,4;M$
9140 NEXT I
9210 DATA "1-WPISANIE  ZRODLOWEG
O"
9220 DATA "2-DOPISANIE ZRODLOWEG
O"
9230 DATA "3-ZAPISANIE ZRODLOWEG
O"
9240 DATA "4-ZAPISANIE WYNIKOWEG
O"
9250 DATA "5-POWROT DO ZRODLOWEG
O"
9260 DATA "6-POWROT Z  KASOWANIE
M"
9299
9300 REM WYDRUK MAPY
9310 PRINT AT 12,13;"MAPA"
9320 FOR I=14 TO 17
9330 READ M$,A$,B$
9335 LET ST=FN P(VAL A$)
9340 LET DL=FN P(VAL B$)-FN P(VA
L A$)
9350 PRINT AT I,6;M$;ST;",";DL
9360 NEXT I
9370 RETURN
9400 DATA "ZRODLOWY:","36679","3
0054"
9410 DATA "TABLICA :","36696","3
6694"
9420 DATA "WYNIKOWY:","36694","3
6700"
9430 DATA "WOLNE    :","36700","2
3732"
9900 REM AUTOSTART
9910 BORDER 0: PAPER 0: INK 7
9920 CLEAR 29999: LET N$="#GENS3
"
9930 LET ST=30000: LET DL=8354
9940 GO SUB 500
9950 BEEP 1,1: RANDOMIZE USR 300
00
9960 POKE 23609,50: POKE 23658,8
9970 RUN
9979
9998 SAVE *"GENS3" LINE 9900
9999 SAVE *"#GENS3"CODE 30000,83
54
```

DYSK TWARDY WINCHESTER

Dla każdego, kto choć trochę interesował się bronią strzelecką lub historią Dzikiego Zachodu nazwa Winchester wydaje się znajoma. Tak nazywały się znakomite karabiny samopowtarzalne produkowane od 1866 r. w założonej przez Olivera F. Winchestera wytwórni broni w New Heaven. Ponad sto lat po sukcesie karabinów samopowtarzalnych podobnym powodzeniem na światowych rynkach zaczął się cieszyć inny wyrób *Made in USA*, noszący tę samą nazwę, lecz przeznaczony do zgoła odmiennych zastosowań.

Jest to opracowana w latach siedemdziesiątych w laboratoriach International Business Machines stacja dysku twardego dla komputerów osobistych. Jakie zalety zadecydowały o sukcesie twardych dysków? W stosunku do typowej stacji dysku elastycznego dysk twardy charakteryzuje się znacznie większą pojemnością (dysk elastyczny dla IBM PC/XT 360 KB, dla IBM PC/AT 1,2 MB, dysk twardy co najmniej 10 MB), a także krótszym czasem dostępu do informacji.

Użytkownikowi siadającemu do klawiatury nie znanego mu IBM-a trudno jest nawet w pierwszej chwili stwierdzić, czy komputer wyposażony jest w dysk typu Winchester. Zazwyczaj gotowość dysku do pracy potwierdza świecąca się na płycie czołowej komputera lampka kontrolna (dioda LED). Praca dysku twardego jest praktycznie bezgłośna (wytwarzany przez niego cichy szum jest nieporównywalny z hałasem wytwarzanym przez typową stację dyskietek 5,25 cala).

O ile o stacji dysku elastycznego można powiedzieć, że stanowi ona solidny produkt współczesnej mechaniki precyzyjnej i elektroniki, to materiały, precyzja wykonania i rozwiązania techniczne dysków typu Winchester zmuszają do zakwalifikowania ich do grupy wyrobów produkowanych z wykorzystaniem „kosmicznych" technologii.

Zasadniczym elementem dysku twardego jest tzw. pakiet dyskowy. Składa się on z kilku (od trzech do ośmiu) dysków wykonanych ze stopu aluminiowego, osadzonych na wspólnej osi. Każda z płytek powleczona jest z obu stron warstwą nośnika z ferromagnetycznych tlenków o grubości zaledwie kilku mikrometrów. Oś pakietu połączona jest bezpośrednio z osią elektrycznego silnika prądu stałego. Pakiet wiruje wraz z osią silnika z prędkością obrotową wynoszącą ok. 3000 obr./min. Z każdym dyskiem należącym do pakietu współpracują dwie głowice, z których jedna umieszczona jest nad górną a druga nad dolną powierzchnią dysku. Każda z głowic zamocowana jest na ramieniu połączonym z osią obrotu. Wszystkie głowice poruszają się razem, wzdłuż promienia pakietu dzięki napędzającemu oś obrotu silnikowi krokowemu. Na jednej z powierzchni dyskowych zapisane są dane dotyczące numerów identyfikacyjnych ścieżek. Ponieważ głowice poruszają się razem natychmiast po odnalezieniu właściwej ścieżki można przystąpić do odczytania poszukiwanego zapisu, którego należy szukać po prostu na innym „piętrze" pakietu dyskowego.

Jak widać sama zasada działania dysku twardego jest bardzo prosta i w istocie niewiele różni się od zasady działania stacji dysku elastycznego (patrz opis w „InforMiku" 1/1987). Jednak praktyczna realizacja tej prostej idei wymaga zastosowania bardzo nowoczesnych i pomysłowych rozwiązań technicznych. Najpoważniejszym problemem jest zabezpieczenie przed uszkodzeniem cienkiej powierzchni nośnika magnetycznego. Warstwa ferromagnetycznych tlenków, naniesiona na wirującym z dużą prędkością dysku, przy bezpośrednim zetknięciu z nieruchomą głowicą bardzo łatwo może ulec uszkodzeniu. Z drugiej jednak strony wymagane jest zachowanie jak najmniejszej odległości między nośnikiem a głowicą, tak aby zbyt szeroka szczelina powietrzna nie doprowadziła do zaniku zapisywanych i odczytywanych sygnałów lub zbytniego poszerzenia pozostawianego przez głowicę „śladu" magnetycznego decydującego o gęstości zapisu. Zastosowano tu rozwiązanie zapożyczone z techniki budowy... poduszkowców. Wirujący z dużą prędkością dysk wytwarza cienką poduszkę powietrzną, na której unosi się głowica. Szerokość szczeliny powietrznej między głowicą a dyskiem wynosi zaledwie kilka mikrometrów. Oznacza to, że drobina kurzu, odcisk palca czy cząstki dymu tytoniowego mogą skutecznie uniemożliwić zapis i odczyt informacji z dysku.

Problem ten rozwiązano przez zastosowanie technologii montażu z zachowaniem najwyższej czystości i zamknięcie pakietu dyskowego wraz z zespołem głowic we wnętrzu szczelnej obudowy (różnicę ciśnień wewnątrz i na zewnątrz urządzenia wyrównuje się przez specjalny filtr).

W momencie planowanego lub nieplanowanego odłączenia napięcia zasilania od dysku głowice automatycznie cofają się do pozycji spoczynkowej — bezpiecznej dla zapisanej powierzchni nośnika. Całkowite wyeliminowanie możliwości uszkodzenia nośnika magnetycznego pakietu dyskowego jest jednak niemożliwe. Dlatego też po każdorazowym uruchomieniu urządzenia przeprowadzany jest automatyczny test stanu dysku, podczas którego zostają zapamiętane obszary uszkodzone i przy operacjach zapisu i odczytu są one pomijane.

Precyzyjne prowadzenie głowicy nad odczytywaną lub zapisywaną ścieżką na powierzchni dysku zapewnia układ specjalnych serwomechanizmów z cewkami o konstrukcji zbliżonej do cewek głośnikowych. Warto tu wspomnieć, że analogiczną konstrukcję ma układ stabilizacji głowicy optycznej w gramofonie laserowym typu Compact Disc.

Zarządzaniem pracą dysku twardego zajmuje się sterownik umieszczony na standardowej wymiennej karcie. Z punktu widzenia oprogramowania zarządzanie zasobami dysku twardego kontroluje oczywiście dyskowy system operacyjny. Jako ciekawostkę można podać, że możliwy jest podział dysku twardego na obszary obsługiwane przez różne systemy operacyjne np. MS-DOS i XENIX albo CP/M-86.

Na początku lat osiemdziesiątych Winchestery miały pojemność 10—20 MB. Postęp techniczny w tej dziedzinie jest na tyle szybki, że obecnie są już w handlu pamięci tego typu o pojemności o rząd wielkości większej. Na przykład w komputerze osobistym Compaq DESKPRO 386/20 zainstalowany jest Winchester 300 MB, zaś w jego przenośnej wersji Compaq PORTABLE 386/20 pojemność dysku twardego liczy 130 MB. Zmieniają się również gabaryty urządzenia. Pierwotnie Winchestery miały rozmiary zbliżone do stacji dysków elastycznych 5,25 cala. Najnowsze rozwiązania pozwalają na instalowanie napędów dysków twardych wprost na standardowych kartach formatu IBM PC, na których znajduje się zarówno sterownik jak i sam napęd dyskowy ze zminiaturyzowanym pakietem o średnicy 3,5 cala („InforMik" pisał o tym w numerze 2/1987). Miniaturyzacja pakietów stała się możliwa dzięki wzrostowi gęstości zapisu, poprawie precyzji ustawienia głowic, a także stosowaniu nowych materiałów na nośniki ferromagnetyczne. Wzrasta poziom niezawodności urządzeń, a średni czas między awariami sięga już dziesiątek tysięcy godzin. Nowe konstrukcje są w większym stopniu uodpornione na niekorzystne warunki eksploatacji — przede wszystkim na wstrząsy.

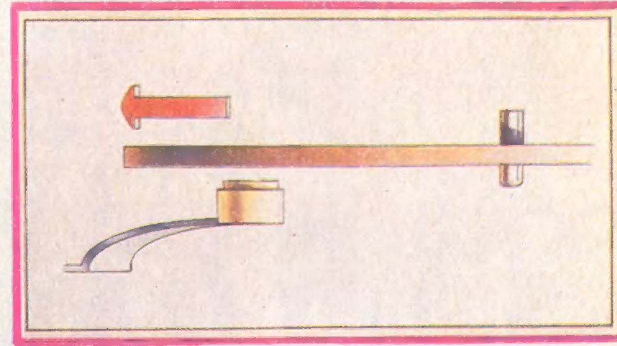
Jacek Nowicki **Krzysztof Zięcina**

„Młody Technik — InforMik" jest wydawany przez Instytut Wydawniczy „Nasza Księgarnia"

Adres redakcji: ul. Spasowskiego 4, 00-389 Warszawa, lub skr. poczt. 380, 00-950 Warszawa. **Telefony:** centrala 26-24-31 do 36. Dział Łączności z Czytelnikami — w. 60, pozostałe działy w. 42 i 47. Redaktor naczelny: 26-26-27 lub w. 87.
Warunki prenumeraty: ogólnie obowiązujące w kraju, za pośrednictwem urzędów pocztowych. Przy wypełnianiu przekazu na prenumeratę należy podać numer indeksu czasopisma: 366013. W stałej sprzedaży „InforMik" jest w salonie wydawniczym „Naszej Księgarni", ul. Spasowskiego 4A, Warszawa.
Redakcja zastrzega sobie prawo adiustacji i skracania nadesłanych materiałów. Nie zamówionych artykułów, fotografii, rysunków itp. redakcja nie zwraca.
Druk: Zakłady Graficzne w Katowicach. Zam. 0021/4333/8 U-22
Nakład 75 315 egz.

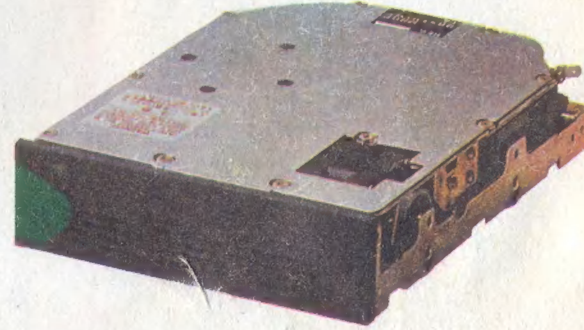
DYSK TWARDY WINCHESTER

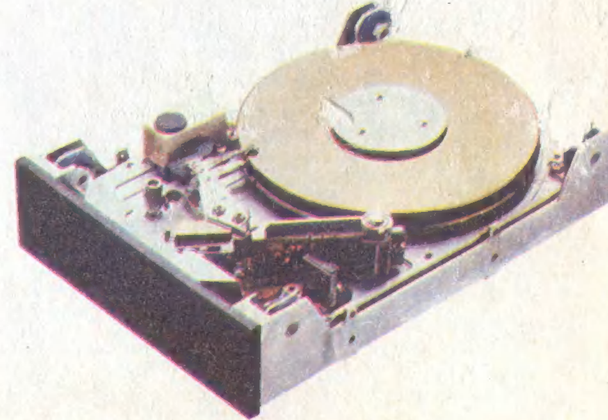
Przekrój jednostki dysku twardego Winchester: 1 — szczelna obudowa pakietu dyskowego, 2 — zespół głowic, 3 — krokowy silnik napędowy ramion z głowicami, 4 — sterownik dysku, 5 — główny silnik napędowy, 6 — obudowa całego urządzenia, 7 — pakiet dyskowy



Głowica magnetyczna dysku twardego typu Winchester unosząca się na poduszce powietrznej

Jednostki dysków twardych 5,25 cala szwajcarskiej firmy Merona. U góry dysk o pojemności 40 MB, poniżej 20 MB





Cena: zł 120,— Indeks nr 366013 PL ISSN 0860-5696